

# The solvability of consensus in iterated models extended with safe-consensus<sup>\*†</sup>

Rodolfo Conde  
rodolfo@matem.unam.mx

Sergio Rajsbaum  
rajsbaum@im.unam.mx

Instituto de Matemáticas, Universidad Nacional Autónoma de México  
Ciudad Universitaria, México D.F. 04510, México

## Abstract

The safe-consensus task was introduced by Afek, Gafni and Lieber (DISC'09) as a weakening of the classic consensus. When there is concurrency, the consensus output can be arbitrary, not even the input of any process. They showed that safe-consensus is equivalent to consensus, in a wait-free system. We study the solvability of consensus in three shared memory iterated models extended with the power of safe-consensus black boxes. In the first model, for the  $i$ -th iteration, processes write to the memory, invoke safe-consensus boxes and finally they snapshot the memory. We show that in this model, any wait-free implementation of consensus requires  $\binom{n}{2}$  safe-consensus black-boxes and this bound is tight. In a second iterated model, the processes write to memory, then they snapshot it and finally they invoke safe-consensus boxes. We prove that in this model, consensus cannot be implemented. In the last iterated model, processes first invoke safe-consensus, then they write to memory and finally they snapshot it. We show that this model is equivalent to the previous model and thus consensus cannot be implemented.

**Keywords:** Consensus, safe-consensus, coalition, Johnson graph, connectivity, distributed algorithms, lower bounds, wait-free computing, iterated models.

---

<sup>\*</sup>A Preliminar version of these results appeared in SIROCCO 2014.

<sup>†</sup>Preliminary version, submitted for publication to a journal.

# 1 Introduction

The ability to agree on a common decision is key to distributed computing. The most widely studied agreement abstraction is *consensus*. In the consensus task each process proposes a value, and all correct processes have to decide the same value. In addition, *validity* requires that the decided value is a proposed value.

Herlihy’s seminal paper [27] examined the power of different synchronization primitives for *wait-free computation*, e.g., when computation completes in a finite number of steps by a process, regardless of how fast or slow other processes run, and even if some of them halt permanently. He showed that consensus is a universal primitive, in the sense that a solution to consensus (with read/write registers) can be used to implement any synchronization primitive in a wait-free manner. Also, consensus cannot be wait-free implemented from read/write registers alone [22, 33]; indeed, all modern shared-memory multiprocessors provide some form of universal primitive.

Afek, Gafni and Lieber [2] introduced *safe-consensus*, which seemed to be a synchronization primitive much weaker than consensus. The validity requirement becomes: if the first process to invoke the task returns before any other process invokes it, then it outputs its input; otherwise, when there is concurrency, the consensus output can be arbitrary, not even the input of any process. In any case, all processes must agree on the same output value. Trivially, consensus implements safe-consensus. Surprisingly, they proved that the converse is also true, by presenting a wait-free implementation of consensus using safe-consensus black-boxes and read/write registers. Why is it then, that safe-consensus seems a much weaker synchronization primitive?

**Our Results.** We show that while consensus and safe-consensus are wait-free equivalent, any wait-free implementation of consensus for  $n$  processes requires  $\binom{n}{2}$  safe-consensus black-boxes, and this bound is tight.

Our main result is the lower bound. It uses connectivity arguments based on subgraphs of *Johnson graphs*, and an intricate combinatorial and bivalency argument, that yields a detailed bound on how many safe-consensus objects of each type (fan-in) are used by the implementation protocol. For the upper bound, we present a simple protocol, based on the new *g-2coalitions-consensus* task, which may be of independent interest<sup>1</sup>.

We develop our results in an iterated model of computation [35], where the processes repeatedly: write their information to a (fresh) shared array, invoke (fresh) safe-consensus boxes and snapshot the contents of the shared array.

Also, we study the solvability of consensus in two alternate iterated models extended with safe-consensus. In the first model, the processes write to memory, then they snapshot it and finally they invoke safe-consensus boxes. We prove that in this model, consensus cannot be implemented from safe-consensus. In the second model, processes first invoke safe-consensus, then they write to shared memory and finally they snapshot the contents of the memory. We show that this model is equivalent to the previous model and thus consensus cannot be solved in this model.

**Related Work.** Distributed computing theory has been concerned from early on with understanding the relative power of synchronization primitives. The wait-free context is the basis to study other failure models e.g. [9], and there is a characterization of the wait-free, read/write solvable tasks [31]. For instance, the weakening of consensus, *set agreement*, where  $n$  processes may agree on at most  $n - 1$  different input values, is still not wait-free solvable [10, 31, 38] with read/write

---

<sup>1</sup>These results appeared for the first time in the Proceedings of the 21st International Colloquium on Structural Information and Communication Complexity [20].

registers only. The *renaming* task where  $n$  processes have to agree on at most  $2n - 1$  names has also been studied in detail e.g. [4, 13, 15, 16, 17].

Iterated models e.g. [12, 29, 30, 35, 36, 37] facilitate impossibility results, and (although more restrictive) facilitate the analysis of protocols [25]. We follow in this paper the approach of [26] that used an iterated model to prove the separation result that set agreement can implement renaming but not vice-versa, and expect our result can be extended to a general model model using simulations, as was done in [24] for that separation result. For an overview of the use of topology to study computability, including the use of iterated models and simulations see [28].

Afek, Gafni and Lieber [2] presented a wait-free protocol that implements consensus using  $\binom{n}{2}$  safe-consensus black-boxes (and read/write registers). Since our implementation uses the weaker, iterated form of shared-memory, it is easier to prove correct. Safe-consensus was used in [2] to show that the  $g$ -tight-group-renaming task [3] is as powerful as  $g$ -consensus.

The idea of the classical consensus impossibility result [22, 33] is (roughly speaking) that the executions of a protocol in such a system can be represented by a graph which is always connected. The connectivity invariance has been proved in many papers using the critical state argument introduced in [22], or sometimes using a layered analysis as in [34]. Connectivity can be used also to prove time lower bounds e.g. [5, 21, 34]. We extend here the layered analysis to prove the lower bound result on the number of safe-consensus objects needed to implement consensus. Also, our results show that when the basic shared memory iterated model is used with objects stronger than read/write memory, care has to be taken in the way they are added to the model, as the resulting power of the model to solve tasks can vary.

In a previous work [19] we had already studied an iterated model extended with the power of safe-consensus. However, that model had the restriction that in each iteration, all processes invoke the same safe-consensus object. We showed that set agreement can be implemented, but not consensus. The impossibility proof uses much simpler connectivity arguments than those of this paper.

The paper is organized as follows. Section 2 describes the basic concepts and previous results of the models of computation. This section can be skipped by readers familiar with standard distributed computing notions. Section 3 defines the three iterated models of computation that we investigate and also we present the main results for each model. Section 4 is devoted to develop all the results obtained for our first iterated model, in it the processes write to memory, invoke the safe-consensus objects and then they snapshot the shared memory. For this model, our results are the following:

- We construct a protocol that solves  $n$ -process consensus using  $\binom{n}{2}$  safe-consensus boxes (Section 4.1). We give our consensus protocol using the new 2coalitions-consensus task.
- We describe for the case of three processes the main result for this iterated model, which is also the main result of this paper, it is a lower bound on the number of safe-consensus objects needed to solve consensus in this iterated model (Section 4.2). The 3-process case illustrates some of the main ideas of the general case. The full proof of the lower bound for  $n$  processes is given in the Appendices.

Section 5 is dedicated to our second iterated model with safe-consensus. In this model, the processes first write to memory, then they snapshot the contents of the memory and after that, they invoke safe-consensus objects. We prove that in this model, consensus cannot be implemented. In Section 6, we develop our last iterated model extended with safe-consensus. In this model, the processes

first invoke safe-consensus objects, then they write to the memory and finally, they snapshot the shared memory. We prove that this model is equivalent to the previous model (for task solvability), thus the consensus task cannot be implemented in this model. Section 7 contains our conclusions and some open problems.

## 2 Basic definitions

In this section, we introduce the model of computation and present many basic concepts used in this paper. We follow the usual definitions and extend some concepts from [6, 7].

### 2.1 Distributed systems

Our formal model is an extension of the standard iterated version [12] of the usual read/write shared memory model e.g. [7]. A *process* is a deterministic state machine, which has a (possible infinite) set of *local states*, including a subset called the *initial states* and a subset called the *output states*.

A *shared object*  $O$  has a domain  $D$  of input values, and a domain  $D'$  of output values.  $O$  provides a unique operation,  $O.exec(d)$ , that receives an input value  $d \in D$  and returns instantaneously an output value  $d' \in D'$ .

A one-shot snapshot object  $S$  is a shared memory array  $S[1, \dots, n]$  with one entry per process. That array is initialized to  $[\perp, \dots, \perp]$ , where  $\perp$  is a default value that cannot be written by a process. The snapshot object  $S$  provides two atomic operations that can be used by a process at most once:

- $S.update(v)$ : when called by process  $p_j$ , it writes the value  $v$  to the register  $S[j]$ .
- $S.scan()$ : returns a copy of the whole shared memory array  $S$ .

It is also customary to make no assumptions about the size of the registers of the shared memory, and therefore we may assume that each process  $p_i$  can write its entire local state in a single register. Notice that the snapshot operation can be implemented in read/write shared memory, according to [1, 11].

A *system* consists of the following data:

- A set of  $n \geq 1$  processes  $\Pi = \{p_1, \dots, p_n\}$ ;
- a shared memory  $SM[i]$  ( $i \geq 0$ ) structured as an infinite sequence of snapshot objects;
- an infinite sequence  $S[j]$  ( $j \geq 0$ ) of shared objects.

A *global state* of a system is a vector  $S$  of the form

$$S = \langle s_1, \dots, s_n; SM \rangle,$$

where  $s_i$  is the local state of process  $p_i \in \Pi$  and  $SM$  is the shared memory of the system. An *initial state* is a state in which every local state is an initial local state and all register in the shared memory are set to  $\perp$ . A *decision state* is a state in which all local states are output states. When referring to a global state  $S$ , we usually omit the word global and simply refer to  $S$  as a state.

## 2.2 Events and round schedules

The occurrences that can take place in a system are modeled as *events*. An *event* in the system is performed by a single process  $p_i \in \Pi$ , which applies only one of the following actions: a write (W) or read (R) operation on the shared memory or an invocation to a shared object (S). Any of these operations may be preceded/followed by some local computation, formally a change of the process to its next local state. We will need to consider events performed *concurrently* by the processes. If  $E$  is any event and  $p_{i_1}, \dots, p_{i_k} \in \Pi$  are processes, then we denote the fact that  $p_{i_1}, \dots, p_{i_k}$  execute concurrently the event  $E$  by  $E(X)$ , where  $X = \{i_1, \dots, i_k\}$ .

We fix once and for all some notation. Let  $\bar{n} = \{1, \dots, n\}$ , when convenient, we will denote  $E(X)$  by  $E(i_1, \dots, i_k)$  and if  $i \in \bar{n}$  is a process id, then  $E(\bar{n} - \{i\})$  is written simply as  $E(\bar{n} - i)$ .

A *round schedule*  $\pi$  is a finite sequence of events of the form

$$\pi: E_1(X_1), \dots, E_r(X_r),$$

that encodes the way in which the processes with ids in the set  $\bigcup_{j=1}^r X_j$  perform the events  $E_1, \dots, E_r$ . For example, the round schedule given by

$$W(1, 3), R(1, 3), W(2), R(2), S(1, 2, 3)$$

means that processes  $p_1, p_3$  perform the write and read events concurrently; after that,  $p_2$  executes in solo its read and write events and finally all three processes invoked the shared objects concurrently. Similarly, the round schedule  $W(1, 2, 3), R(1, 2, 3), S(1, 2, 3)$  says that  $p_1, p_2$  and  $p_3$  execute concurrently the write and read events in the shared memory and then they invoked the shared objects concurrently.

## 2.3 Protocols and executions

The state machine of each process  $p_i \in \Pi$  is called a *local protocol*  $\mathcal{A}_i$ , that determines the steps taken by  $p_i$ . We assume that all local protocols are identical; i.e. Processes have the same state machine. A *protocol* is a collection  $\mathcal{A}$  of local protocols  $\mathcal{A}_1, \dots, \mathcal{A}_n$ .

For the sake of simplicity, we will give protocols specifications using pseudocode and we establish the following conventions: A lowercase variable denotes a local variable, with a subindex that indicates to which process it belongs; the shared memory (which is visible to all processes) is denoted with uppercase letters. Intuitively, the local state  $s_i$  of process  $p_i$  is composed of the contents of all the local variables of  $p_i$ . Also, we identify two special components of each process' states: an input and an output. It is assumed that initial states differ only in the value of the input component; moreover, the input component never changes. The protocol cannot overwrite the output, it is initially  $\perp$ ; once a non- $\perp$  value is written to the output component of the state, it never changes; when this occurs, we say that the process *decides*. The output states are those with non- $\perp$  output values.

Let  $\mathcal{A}$  be a protocol. An *execution* of  $\mathcal{A}$  is a finite or infinite alternating sequence of states and round schedules

$$S_0, \pi_1, \dots, S_k, \pi_{k+1}, \dots,$$

where  $S_0$  is an initial state and for each  $k \geq 1$ ,  $S_k$  is the resulting state of applying the sequence of events performed by the processes in the way described by the round schedule  $\pi_k$ . An *r-round partial execution* of  $\mathcal{A}$  is a finite execution of  $\mathcal{A}$  of the form  $S_0, \pi_1, \dots, S_{r-1}, \pi_r, S_r$ .

If  $P$  is a state,  $P$  is said to be *reachable in  $\mathcal{A}$*  if there exists an  $r$ -round partial execution of  $\mathcal{A}$  ( $r \geq 0$ ) that ends in the state  $P$  and when there is no confusion about which protocol we refer to, we just say that  $S$  is reachable.

Given the protocol  $\mathcal{A}$  and two states  $S, R$ , we say that  $R$  is a *successor* of  $S$  in  $\mathcal{A}$ , if there exists an execution  $\alpha$  of  $\mathcal{A}$  such that

$$\alpha = S_0, \pi_1, \dots, S_r = S, \pi_{r+1}, \dots, \pi_{r+k}, S_{r+k} = R, \dots,$$

i.e., starting from  $S$ , we can run the protocol  $\mathcal{A}$   $k$  rounds (for some  $k \geq 0$ ) such that the system enters state  $R$ . If  $\pi$  is any round schedule and  $S$  is a state, the successor of  $S$  in  $\mathcal{A}$  obtained by running the protocol (starting in the state  $S$ ) one round with the round schedule  $\pi$  is denoted by  $S \cdot \pi$ .

## 2.4 Decision tasks

In distributed computing, a *decision task* is a problem that must be solved in a distributed system. Each process starts with a private input value, communicates with the others, and halts with a private output value. Formally, a *decision task*  $\Delta$  is a relation that has a domain  $\mathcal{I}$  of input values and a domain  $\mathcal{O}$  of output values;  $\Delta$  specifies for each assignment of the inputs to processes on which outputs processes can decide. A *bounded* decision task is a task whose number of input values is finite.

We also refer to decision task simply as tasks. Examples of tasks includes *consensus* [22], *renaming* [3, 11] and the *set agreement* task [18].

A protocol  $\mathcal{A}$  *solves* a decision task  $\Delta$  if any finite execution  $\alpha$  of  $\mathcal{A}$  can be extended to an execution  $\alpha'$  in which all processes decide on values which are allowable (according to  $\Delta$ ) for the inputs in  $\alpha$ . Because the outputs cannot be overwritten, if a process has decided on a value in  $\alpha$ , it must have the same output in  $\alpha'$ . This means that outputs already written by the processes can be completed to outputs for all processes that are permissible for the inputs in  $\alpha$ .

A protocol  $\mathcal{A}$  is *wait-free* if in any execution of  $\mathcal{A}$ , a process either it has a finite number of events or it decides. This implies that if a process has an infinite number of events, it must decide after a finite number of events. Roughly speaking,  $\mathcal{A}$  is wait-free if any process that continues to run will halt with an output value in a fixed number of steps, regardless of delays or failures by other processes. However, in our formal model, we do not require the processes to halt; they solve the decision task and decide by writing to the output component; processes can continue to participate. We typically consider the behavior of a process until it decides, and therefore, the above distinction does not matter.

The study of wait-free shared memory protocols has been fundamental in distributed computing, some of the most powerful results have been constructed on top of wait-free protocols [10, 31, 35, 38]. Also, other variants of distributed systems can be reduced to the wait-free case [9, 10, 23].

### Definition of consensus and safe-consensus tasks.

The tasks of interest in this paper are the *consensus* and *safe-consensus* [2] tasks.

**Consensus** Every process starts with some initial input value taken from a set  $I$  and must output a value such that:

- Termination: Each process must eventually output some value.
- Agreement: All processes output the same value.
- Validity: If some process outputs  $v$ , then  $v$  is the initial input of some process.

**Safe-consensus** Every process starts with some initial input value taken from a set  $I$  and must output a value such that Termination and Agreement are satisfied, and:

- Safe-Validity: If a process  $p_i$  starts executing the task and outputs before any other process starts executing the task, then its decision is its own proposed input value. Otherwise, if two or more processes access the safe-consensus task concurrently, then any decision value is valid.

The safe-consensus task [2] is the result of weakening the validity condition of consensus.

### 3 Iterated models extended with safe-consensus

Intuitively, a model of distributed computing describes a set of protocols that share some common properties and/or restrictions in the way the processes can access the shared objects and these restrictions affect the way in which the protocols can be specified. In this paper, we are interested in protocols which can be written in a simple and structured way, such that the behavior of the system in the  $i$ th-iteration, can be described by using the behavior of the  $(i - 1)$ th-iteration, in an inductive way.

In this section, we introduce an extension of the basic iterated model of [12], adding the power of safe-consensus shared objects. We also present all the results of this paper.

#### 3.1 The iterated model with shared objects

In the iterated model extended with shared objects, the processes can use two kinds of communication media. The first is the shared memory  $SM$  structured as an infinite array of snapshot objects; the second medium is the infinite array  $S$  of shared objects ( $SM$  and  $S$  are described in Section 2.1). The processes communicate between them through the snapshot objects and the shared objects of  $S$ , in an asynchronous and round-based pattern. In all the iterated models that we investigate, we make the following assumptions:

- The operations **update** and **scan** of the snapshot objects in  $SM$  can be used by a process at most once.
- The *exec* operation of each shared object in  $S$  can be used at most once by each process that invokes it.

When we want to add the power of shared objects to the standard iterated model [12], we must consider two important questions. The first question is: In which order should we place the three basic operations (write, read and invoke a shared object) ? We have three possibilities:

- Write, read from the shared memory and invoke a shared object;

- Invoke a shared object, write and read the shared memory;
- Write to memory, invoke a shared object and read the contents of the shared memory.

The second question, which is very closely related to the previous one is: Does the order of the main operations affect the computational power of the new model (for task solvability) ? In this paper, we address these two questions and show the differences of the models of distributed computing that we obtain when the shared objects invoked by the processes are safe-consensus objects.

A *safe-consensus object* is a shared object that can be invoked by any number of processes. The object receives an input value from each process that invokes it, and returns to all the processes an output value that satisfies the Agreement and Validity condition of the safe-consensus task. In other words, a safe-consensus object is like a “black box” that the processes can use to solve instances of the safe-consensus task. The method of using distributed tasks as black boxes inside protocols is a standard way to study the relative computational power of distributed tasks (i.e. if one task is weaker than another, see [2, 14, 26]). Notice that safe-consensus shared objects are primitives more powerful than the read/write shared memory registers, as the safe-consensus task can implement consensus [2].

From now on, we work exclusively in iterated models, where the shared objects invoked by the processes are safe-consensus objects.

### 3.2 The WOR iterated model

We now define the first iterated model that we investigate; in it, processes write to shared memory, then they invoke safe-consensus objects and finally they snapshot the shared memory. A protocol  $\mathcal{A}$  is a protocol in the *WOR* (*Write, invoke Object and Read*) iterated model if it can be written as specified in Figure 1.

```

(1) init  $r \leftarrow 0$ ;  $sm \leftarrow input$ ;  $dec \leftarrow \perp$ ;  $val \leftarrow \perp$ ;

(2) loop forever
(3)    $r \leftarrow r + 1$ ;
(4)    $SM[r].update(sm, val)$ ;
(5)    $val \leftarrow S[h(\langle r, id, sm, val \rangle)].exec(v)$ ;
(6)    $sm \leftarrow SM[r].scan()$ ;

(7)   if ( $dec = \perp$ ) then
(8)      $dec \leftarrow \delta(sm, val)$ ;
(9)   end if
(10) end loop

```

Figure 1: The WOR iterated model

An explanation of the pseudocode in Figure 1 follows. All the variables  $r, sm, val, input$  and  $dec$  are local to process  $p_i$  and only when we analyze a protocol, we add a subindex  $i$  to a variable to specify it is local to  $p_i$ . The symbol “ $id$ ” contains the executing process’ id. Initially,  $r$  is zero and  $sm$  is assigned the contents of the readonly variable  $input$ , which contains the input value for process  $p_i$ ; all other variables are initialized to  $\perp$ . In each round,  $p_i$  increments by one the loop counter  $r$ , accesses the current shared memory array  $SM[r]$ , writing all the information it



has stored in  $sm$  and  $val$  (full information) and then  $p_i$  decides which shared object it is going to invoke by executing a deterministic function  $h$  that returns an index  $l$ , then  $p_i$  invokes the shared object  $S[l]$  with some value  $v$ . Then,  $p_i$  takes a snapshot of the shared array and finally,  $p_i$  checks if  $dec$  is equal to  $\perp$ , if so, it executes a deterministic function  $\delta$  to determine if it may *decide* a valid output value or  $\perp$ . Notice that in each round of a protocol, each process invokes at most one safe-consensus object of the array  $S$ .

It turns out that the WOR iterated model is quite different from the two other iterated models. This is true because of the following facts:

- The consensus problem for  $n$  processes can be solved in the WOR iterated model using only  $\binom{n}{2}$  safe-consensus black boxes (Theorem 4.1).
- Any protocol in the WOR iterated model which implements consensus using safe-consensus objects must use at least  $\binom{n}{2}$  safe-consensus objects.

The second fact, which is a consequence of Theorem 4.2, is the main result of this paper. It describes a matching lower bound on the number of safe-consensus objects needed to solve consensus by any protocol in the WOR iterated model which implements consensus. In Section 4, we give the detailed description of the WOR iterated protocol which implements consensus using safe-consensus objects, we prove its correctness and finally, we give the proof of the lower bound on the number of safe-consensus objects needed to solve consensus in the WOR iterated model.

### 3.3 The WRO iterated model

The second iterated model that we study is the *WRO (Write-Read, invoke Object) iterated model*. In this model, processes write to shared memory, then they snapshot it, and finally they invoke the safe-consensus object(s). We say that a protocol is in the WRO iterated model if it can be written in the form given in Figure 2.

```

(1) init  $r \leftarrow 0$ ;  $sm \leftarrow input$ ;  $dec \leftarrow \perp$ ;  $val \leftarrow \perp$ ;
(2) loop forever
(3)    $r \leftarrow r + 1$ ;
(4)    $SM[r].update(sm, val)$ ;
(5)    $sm \leftarrow SM[r].scan()$ ;
(6)    $val \leftarrow S[h(\langle r, id, sm, val \rangle)].exec(v)$ ;
(7)   if ( $dec = \perp$ ) then
(8)      $dec \leftarrow \delta(sm, val)$ ;
(9)   end if
(10) end loop

```

Figure 2: The WRO iterated model

This pseudocode is explained in a similar way to that used for the code in Figure 1, the only thing that changes is the place where we put the invocations to the safe-consensus shared objects, after the execution of the write-snapshot operations.

In Section 5, we prove that the consensus task cannot be implemented in the WRO iterated model using safe-consensus objects (Theorem 5.4), this is the main result for this iterated model.

### 3.4 The OWR iterated model

The last iterated model that we introduce is constructed by placing the safe-consensus objects before the write and snapshot operations. A protocol  $\mathcal{A}$  is in the *OWR (invoke Object, Write-Read) iterated model* if  $\mathcal{A}$  can be written in the form specified in Figure 3.

```

(1) init  $r \leftarrow 0$ ;  $sm \leftarrow input$ ;  $dec \leftarrow \perp$ ;  $val \leftarrow \perp$ ;
(2) loop forever
(3)    $r \leftarrow r + 1$ ;
(4)    $val \leftarrow S[h(\langle r, id, sm, val \rangle)].exec(v)$ ;
(5)    $SM[r].update(sm, val)$ ;
(6)    $sm \leftarrow SM[r].scan()$ ;
(7)   if ( $dec = \perp$ ) then
(8)      $dec \leftarrow \delta(sm, val)$ ;
(9)   end if
(10) end loop

```

Figure 3: The OWR iterated model

In Section 6, we prove that for task solvability, there is no real difference between the WRO and the OWR iterated models. Any protocol in the WRO iterated model can be simulated by a protocol in the OWR iterated model and the converse is also true, this is stated formally in Theorem 6.4. Combining this result with Theorem 5.4, we can conclude that it is impossible to solve consensus in the OWR iterated model (Corollary 6.5).

### 3.5 Shared objects represented as combinatorial sets

We now introduce some combinatorial definitions which will help us represent shared objects and the specific way in which the processes can invoke these shared objects. These definitions are useful in Sections 5 and 4.

For any  $n \geq 1$  and  $m \in \bar{n}$ , let  $V_{n,m} = \{c \subseteq \bar{n} \mid |c| = m\}$ . Given a protocol  $\mathcal{A}$ , we define for each  $m \leq n$  the set  $\Gamma_{\mathcal{A}}(n, m) \subseteq V_{n,m}$  as follows:  $b = \{i_1, \dots, i_m\} \in \Gamma_{\mathcal{A}}(n, m)$  if and only if in some iteration of the protocol  $\mathcal{A}$ , only the processes  $p_{i_1}, \dots, p_{i_m}$  invoke a safe-consensus object of the array  $S$  (see Figures 1, 2 and 3). Roughly speaking, each  $c \in \Gamma_{\mathcal{A}}(n, m)$  represents a set of processes which together can invoke safe-consensus shared objects in  $\mathcal{A}$ .

For example, if  $m = 3$  and  $c = \{i, j, k\} \in \Gamma_{\mathcal{A}}(n, 3)$ , then in at least one round of  $\mathcal{A}$ , processes  $p_i, p_j$  and  $p_k$  invoke a safe-consensus object and if in other iteration or perhaps another execution of  $\mathcal{A}$ , these processes invoke another safe-consensus object in the same way, then these two invocations are represented by the same set  $c \in \Gamma_{\mathcal{A}}(n, 3)$ , that is, shared objects invoked by the same processes are considered as the same element of  $\Gamma_{\mathcal{A}}(n, 3)$  (repetitions do not count). On the other hand, if  $d = \{i, j, l\} \notin \Gamma_{\mathcal{A}}(n, 3)$ , then there does not exist an execution of  $\mathcal{A}$  in which only the three processes  $p_i, p_j$  and  $p_l$  invoke a safe-consensus shared object. For the consensus protocol of Section 4.1, we have that for  $n = 4$ ,  $\Gamma_{\mathcal{A}}(4, 2) = \{\{1, 2\}, \{2, 3\}, \{3, 4\}\}$ ,  $\Gamma_{\mathcal{A}}(4, 3) = \{\{1, 2, 3\}, \{2, 3, 4\}\}$  and  $\Gamma_{\mathcal{A}}(4, 4) = \{\{1, 2, 3, 4\}\}$ . A set  $b \in \Gamma_{\mathcal{A}}(n, m)$  is called a *m-box* or simply a *box*. An element  $d \in \Gamma_{\mathcal{A}}(n, 1)$  is called a *trivial box*, it represents a safe-consensus object invoked only by one process, we consider such invocations as useless, because they do not give any additional information to the process. We model a process that does not invoke a safe-consensus object as a process that invokes

a safe-consensus object and no other process invokes that object, i.e., this safe-consensus object is represented by a trivial box. A process  $p_i$  *participates* in the box  $b$  if  $i \in b$ . Let the set  $\Gamma_{\mathcal{A}}(n)$  and the quantities  $\nu_{\mathcal{A}}(n, m)$  and  $\nu_{\mathcal{A}}(n)$  be defined as follows:

$$\Gamma_{\mathcal{A}}(n) = \bigcup_{m=2}^n \Gamma_{\mathcal{A}}(n, m);$$

$$\nu_{\mathcal{A}}(n, m) = |\Gamma_{\mathcal{A}}(n, m)|;$$

$$\nu_{\mathcal{A}}(n) = \sum_{m=2}^n \nu_{\mathcal{A}}(n, m).$$

From now on, for all our protocols, we consider global states only at the end of some iteration. Suppose that  $S$  is a reachable state in the protocol  $\mathcal{A}$ . The set of shared objects  $o_1, \dots, o_q$  invoked by the processes to enter the state  $S$  is represented by a set of boxes  $\mathbf{Inv}(S) = \{b_1, \dots, b_q\}$  which is called the *global invocation specification* of  $S$ . We assume without loss of generality that in all rounds, each process invokes some shared object, that is, the set  $\mathbf{Inv}(S)$  satisfies

$$\bigcup_{b \in \mathbf{Inv}(S)} b = \bar{n},$$

(a process that does not invoke a safe-consensus object can be seen as a process that invokes a safe-consensus object and no other process invokes that object). If  $b = \{l_1, \dots, l_s\} \in \mathbf{Inv}(S)$  is a box representing a safe-consensus shared object invoked by the processes  $p_{l_1}, \dots, p_{l_s}$ , we define the *safe-consensus value of  $b$  in  $S$* , denoted by  $\text{scval}(b, S)$  as the output value of the safe-consensus shared object represented by  $b$ .

### 3.6 Additional definitions on global states

We now introduce the notions of connectivity and paths between global states. These are well known concepts [22, 31] and have become a fundamental tool to study distributed systems.

#### Paths of global states

Two states  $S, P$  are said to be *adjacent* if there exists a non-empty subset  $X \subseteq \bar{n}$  such that all processes with ids in  $X$  have the same local state in both  $S$  and  $P$ . That is, for each  $i \in X$ ,  $p_i$  cannot *distinguish* between  $S$  and  $P$ . We denote this by  $S \stackrel{X}{\sim} P$ . States  $S$  and  $P$  are *connected*, if we can find a sequence of states (called a *path*)

$$\mathbf{p}: S = P_1 \sim \dots \sim P_r = P,$$

such that for all  $j$  with  $1 \leq j \leq r-1$ ,  $P_j$  and  $P_{j+1}$  are adjacent.

Connectivity of global states are a key concept for many beautiful results in distributed systems, namely, impossibility proof. The indistinguishability of states between processes is the building block to construct topological structures based on the executions of a given protocol and is fundamental in many papers [10, 19, 31, 33, 38]. In addition to the classic definitions of connectivity and paths, we also introduce the following concepts. Let  $\mathbf{q}: Q_1 \sim \dots \sim Q_l$  be a path of connected states, define the *set of states*  $\text{States}(\mathbf{q})$ ; the *set of indistinguishability sets*  $\text{iSets}(\mathbf{q})$ ; and the *degree of indistinguishability*  $\text{deg } \mathbf{q}$ , of  $\mathbf{q}$  as follows:

$$\text{States}(\mathbf{q}) = \{Q_1, \dots, Q_l\};$$

$$\begin{aligned} \text{iSets}(\mathfrak{q}) &= \{X \subseteq \bar{n} \mid (\exists Q_i, Q_j \in \text{States}(\mathfrak{q}))(Q_i \overset{X}{\sim} Q_j)\}; \\ \deg \mathfrak{q} &= \min\{|X| \mid X \in \text{iSets}(\mathfrak{q})\}. \end{aligned}$$

A path  $\mathfrak{s}$  of connected states of  $\mathcal{A}$  is said to be *C-regular* if and only if  $\mathbf{Inv}(S) = \mathbf{Inv}(Q)$  for all  $S, Q \in \text{States}(\mathfrak{s})$ , that is,  $\mathfrak{s}$  is C-regular when all the states in the set  $\text{States}(\mathfrak{s})$  have the same global invocation specification.

**Lemma 3.1.** *Let  $\mathcal{A}$  be an iterated protocol for  $n$  processes,  $A \subseteq \bar{n}$  a non-empty set and  $S, Q$  two reachable states of  $\mathcal{A}$  in round  $r$ , such that for all  $j \in A$ ,  $p_j$  has the same snapshot information in  $S$  and  $Q$ . Then all processes with ids in  $A$  participate in the same boxes in  $S$  and  $Q$ .*

### 3.7 Consensus protocols

We also need some extra definitions regarding consensus protocols: If  $v$  is a valid input value of consensus for processes and  $S$  is a state, we say that  $S$  is *v-valent* if in every execution starting from  $S$ , there exists a process that outputs  $v$ .  $S$  is *univalent* if in every execution starting from  $S$ , processes always outputs the same value. If  $S$  is not univalent, then  $S$  is *bivalent*.

**Lemma 3.2.** *Any two initial states of a protocol for consensus are connected.*

*Proof.* Let  $S, P$  be two initial states. If  $S$  and  $P$  differ only in the initial value  $\text{input}_i$  of a single process  $p_i$ , then  $S$  and  $P$  are adjacent (Only  $p_i$  can distinguish between the two states, the rest of the processes have the same initial values). In the case  $S$  and  $P$  differ in more than one initial value, they can be connected by a sequence of initial states  $S = S_1 \sim \dots \sim S_q = P$  such that  $S_j, S_{j+1}$  differ only in the initial value of some process (we obtain  $S_{j+1}$  from  $S_j$  by changing the input value of some process  $p_i$ , the result is a valid input of the consensus problem), hence they are adjacent. In summary,  $S$  and  $P$  are connected.  $\square$

We need one last result about consensus protocols, we omit its easy proof.

**Lemma 3.3.** *Suppose that  $\mathcal{A}$  is a protocol that solves the consensus task and that  $I, J$  are connected initial states of  $\mathcal{A}$ , such that for all rounds  $r \geq 0$ ,  $I^r, J^r$  are connected successor states of  $I$  and  $J$  respectively. Also, assume that  $I$  is a  $v$ -valent state. Then  $J$  is  $v$ -valent.*

## 4 Solving consensus in the WOR iterated model

In this section, We investigate the solvability of consensus in the WOR iterated model. We first show that there exists a protocol in the WOR iterated model for consensus with  $\binom{n}{2}$  safe-consensus objects. Then we present our main result, the lower bound on the number of safe-consensus objects needed by any protocol in the WOR iterated model which implements consensus. We use terminology and results from Sections 2 and 3. For simplicity, we refer to any protocol in the WOR iterated model as a WOR protocol.

### 4.1 Solving consensus with safe-consensus

In this section, we argue that there exists a WOR protocol that solves the consensus task using precisely  $\binom{n}{2}$  safe-consensus objects. The complete specification of such protocol is in Figure 8, in Appendix A.

A simple way to describe the protocol that solves consensus is by seeing it as a protocol in which the processes use a set of  $\binom{n}{2}$  shared objects which represent an intermediate task which can be implemented using one snapshot object and one safe-consensus object. This task is our new  $g$ -2coalitions-consensus task. It can be defined (roughly) as follows:

**2Coalitions-consensus** We have  $g$  processes  $p_1, \dots, p_g$  and each one starts with some initial input value of the form  $x = \langle v_1, v_2 \rangle$ , where  $v_i \in I \cup \{\perp\}$  such that  $v_1 \neq \perp$  or  $v_2 \neq \perp$ . Let  $x.left$  denote the value  $v_1$  and  $x.right$  the value  $v_2$ . if  $x_1, \dots, x_g$  are the input values of all processes, then it must hold that for all  $i, j$  such that  $x_i.left \neq \perp$  and  $x_j.left \neq \perp$ , then  $x_i.left = x_j.left$ . A similar rule must hold if  $x_i.right \neq \perp$  and  $x_j.right \neq \perp$ . Also, there must exists an unique process with input value  $\langle v, \perp \rangle$  with  $v \neq \perp$  and process  $p_g$  must be the only process with input value  $x_g = \langle \perp, v' \rangle$ , where  $v' \neq \perp$ . Each process must output a value such that Termination and Agreement are satisfied, and:

- 2coalitions-Validity: If some process outputs  $v$ , then there must exists a process  $p_j$  with input  $x_j$  such that  $x_j = \langle v, u \rangle$  or  $x_j = \langle u, v \rangle$  with  $v \in I$ .

Using the task  $g$ -2coalitions-consensus, the protocol in Figure 8 can be described graphically as shown in Figure 4, for the case of  $n = 4$ . In each round of the protocol, some processes invoke a 2coalitions-consensus object, represented by the symbol  $2CC_i$ . In round one,  $p_1$  and  $p_2$  invoke the object  $2CC_1$  with input values  $\langle v_1, \perp \rangle$  and  $\langle \perp, v_2 \rangle$  respectively, (where  $v_i$  is the initial input value of process  $p_i$ ) and the consensus output  $u_1$  of  $2CC_1$  is stored by  $p_1$  and  $p_2$  in some local variables. In round two,  $p_2$  and  $p_3$  invoke the  $2CC_2$  object with inputs  $\langle v_2, \perp \rangle$  and  $\langle \perp, v_3 \rangle$  respectively and they keep the output value  $u_2$  in local variables. Round three is executed by  $p_3$  and  $p_4$  in a similar way, to obtain the consensus value  $u_3$  from the 2coalition-consensus object  $2CC_3$ . At the beginning of round four,  $p_1, p_2$  and  $p_3$  gather the values  $u_1, u_2$  obtained from the objects  $2CC_1$  and  $2CC_2$  to invoke the  $2CC_4$  2coalition-consensus object with the input values  $\langle u_1, \perp \rangle, \langle u_1, u_2 \rangle$  and  $\langle \perp, u_2 \rangle$  respectively (Notice that  $p_2$  uses a tuple with both values  $u_1$  and  $u_2$ ) and they obtain a consensus value  $u_4$ . Similar actions are taken by the processes  $p_2, p_3$  and  $p_4$  in round five with the shared object  $2CC_5$  and the values  $u_2, u_3$  to compute an unique value  $u_5$ . Finally, in round six, all processes invoke the last shared object  $2CC_6$ , with the respective input tuples

$$\langle u_4, \perp \rangle, \langle u_4, u_5 \rangle, \langle u_4, u_5 \rangle, \langle \perp, u_5 \rangle,$$

and the shared object returns to all processes an unique output value  $u$ , which is the decided output value of all processes, thus this is the final consensus of the processes.

The protocol of Figure 5 implements  $g$ -2coalitions-consensus. Each process  $p_i$  receives as input a tuple with values satisfying the properties of the 2coalitions-consensus task and then in lines 3-5,  $p_i$  writes its input tuple in shared memory using the snapshot object  $SM$ ; invokes the safe-consensus object with its id as input, storing the unique output value  $u$  of the shared object in the local variable  $val$  and finally,  $p_i$  takes a snapshot of the memory. Later, what happens in Lines 6-10 depends on the output value  $u$  of the safe-consensus object. If  $u = g$ , then by the Safe-Validity property, either  $p_g$  invoked the object or at least two processes invoked the safe-consensus object concurrently and as there is only one process with input tuple  $\langle v, \perp \rangle$ ,  $p_i$  will find an index  $j$  with  $sm[j].right \neq \perp$  in line 7, assign this value to  $dec$  and in line 11  $p_i$  decides. On the other hand, if  $u \neq g$ , then again by the Safe-Validity condition of the safe-consensus task, either process  $p_u$  is running and invoked the safe-consensus object or two or more processes invoked concurrently the

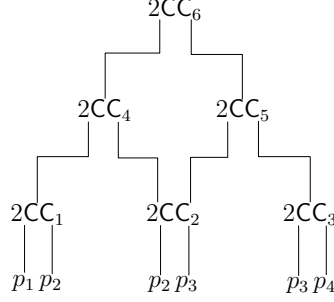


Figure 4: The structure of the 4-consensus protocol using 2coalitions-consensus tasks.

shared object and because all processes with id not equal to  $g$  have input tuple  $\langle z, y \rangle$  with  $z \neq \perp$ , it is guaranteed that  $p_i$  can find an index  $j$  with  $sm[j].left \neq \perp$  and assign this value to  $dec$  to finally execute line 11 to decide its output value. All processes decide the same value because of the properties of the input tuples of the 2coalitions-consensus task and the Agreement property of the safe-consensus task.

```

(1) procedure  $g$ -2coalitions-consensus( $v_1, v_2$ )
(2) begin
(3)    $SM.update(\langle v_1, v_2 \rangle)$ ;
(4)    $val \leftarrow safe-consensus.exec(id)$ ;
(5)    $sm \leftarrow SM.scan()$ ;
(6)   if  $val = g$  then
(7)      $dec \leftarrow$  choose any  $sm[j].right \neq \perp$ ;
(8)   else
(9)      $dec \leftarrow$  choose any  $sm[j].left \neq \perp$ ;
(10)  end if
(11)  decide  $dec$ ;
(12) end

```

Figure 5: A  $g$ -2coalitions-consensus protocol with one safe-consensus object.

**Theorem 4.1.** *There exists a WOR protocol that solves the consensus task for  $n$  processes using  $\binom{n}{2}$  safe-consensus objects.*

The proof of Theorem 4.1 and the correctness proof of the 2coalitions-consensus protocol of Figure 5 are given in Appendix A.

## 4.2 The lower bound

The main result of this paper is a matching lower bound on the number of safe-consensus objects needed to solve consensus using safe-consensus. Our lower bound proof is based partly on standard bivalency arguments [22], but in order to be able to apply them, a careful combinatorial work is necessary. We provide here the full proof for the case of three processes, the general case is described in detail in Appendices B and C.

### Further model terminology.

In all our proofs concerning the WOR iterated model, we assume that the input values that the processes feed to the safe-consensus objects are their own ids (without loss of generality). We now define a set of round schedules that will be very useful for the rest of the paper. Given  $q \geq 1$  disjoint sets  $A_1, \dots, A_q \subset \bar{n}$ , define the round schedule  $\xi(A_1, \dots, A_q, Y)$  for  $\mathcal{A}$  as:

$$W(A_1), S(A_1), R(A_1), \dots, W(A_q), S(A_q), R(A_q), W(Y), S(Y), R(Y), \quad (1)$$

where  $Y = \bar{n} - (\bigcup_{i=1}^q A_i)$ . Sometimes, if there is no confusion, we omit the set  $Y$  and just write  $\xi(A_1, \dots, A_q)$ . For any state  $S$  and  $u \geq 0$ , define

$$S \cdot \xi^u(A_1, \dots, A_q) = \begin{cases} S & \text{if } u = 0, \\ (S \cdot \xi^{u-1}(A_1, \dots, A_q)) \cdot \xi(A_1, \dots, A_q) & \text{otherwise.} \end{cases}$$

I.e.  $S \cdot \xi^u(A_1, \dots, A_q)$  is the state that we obtain after we run the protocol  $\mathcal{A}$  (starting from  $S$ )  $u$  rounds with the round schedule  $\xi(A_1, \dots, A_q)$  in each iteration.

Our lower bound says that any WOR consensus protocol  $\mathcal{A}$  using safe-consensus, must satisfy the inequality  $\nu_{\mathcal{A}}(n) \geq \binom{n}{2}$  for all  $n \geq 2$ . Moreover,  $\mathcal{A}$  also satisfies the inequalities  $\nu_{\mathcal{A}}(n, m) > n - m$  for all  $n \geq 2$  and  $2 \leq m \leq n$ .

### The connectivity of iterated protocols with safe-consensus.

Roughly, a typical consensus impossibility proof shows that a protocol  $\mathcal{A}$  cannot solve consensus because there exist one execution of  $\mathcal{A}$  in which processes decide a consensus value  $v$  and a second execution of  $\mathcal{A}$  where the consensus output of the processes is  $v'$ , with  $v \neq v'$ , such that the global states of these executions can be connected with paths of connected states. The existence of such paths will imply that in some execution of  $\mathcal{A}$ , some processes decide distinct output values [19, 22, 31, 33], violating the Agreement requirement of consensus. Any protocol that solves consensus, must be able to destroy these paths of connected states.

For the case of our lower bound proof, the main circumstance that will prevent  $\mathcal{A}$  from solving consensus is that for some  $m_0 \in \{2, \dots, n\}$ , it is true that  $\nu_{\mathcal{A}}(n, m_0) \leq n - m_0$ , i.e., at most  $n - m_0$  subsets of processes of size  $m_0$  can invoke safe-consensus shared objects in the protocol  $\mathcal{A}$ .

### The lower bound for 3-process consensus protocols.

We are ready to present our main result, it is the following

**Theorem 4.2.** *If  $\mathcal{A}$  is a WOR protocol for  $n$ -consensus using safe-consensus objects, then for every  $m \in \{2, \dots, n\}$ ,  $\nu_{\mathcal{A}}(n, m) > n - m$ .*

Theorem 4.2 describes the minimum number of different ways in which  $m$  processes must invoke safe-consensus shared objects, in order to be able to solve the consensus task, for each  $m = 2, \dots, n$ . The lower bound on the total number of safe-consensus objects necessary to implement consensus, is an easy consequence of Theorem 4.2 and the definition of  $\nu_{\mathcal{A}}(n)$ .

Here we present the full proof of the lower bound, for three processes WOR protocols with safe-consensus. We will show that if  $\mathcal{A}$  is a protocol which solves consensus for three processes, then  $\mathcal{A}$  must satisfy the inequalities

$$\nu_{\mathcal{A}}(3, m) > 3 - m \text{ for } m \in \{2, 3\}.$$

As usual, the proof is by contradiction, we investigate what happens if  $\mathcal{A}$  does not satisfy the given inequalities for some  $m_0$ . We will divide the proof in two cases.

**Case  $m_0 = 2$**  Assume that  $\mathcal{A}$  is a protocol with safe-consensus such that  $\nu_{\mathcal{A}}(3, 2) \leq 3 - 2 = 1$ , that is, at most two fixed processes can invoke together safe-consensus shared objects. To begin, we need a small combinatorial result.

**Lemma 4.3.** *Let  $U \subset V_{n,2}$  with  $|U| \leq n - 2$ . Then there exists a partition  $\bar{n} = A \cup B$  such that*

$$(\forall b \in U)(b \subseteq A \text{ or } b \subseteq B). \quad (2)$$

The previous result can be proven using subgraphs of *Johnson graphs*. Lemma 4.3 will help us to prove Lemma 4.6, which is a structural result that we use to construct a bivalency argument to show the lower bound for the case of  $m_0 = 2$  in the proof of Theorem 4.2.

**Lemma 4.4.** *Let  $S$  be a state of  $\mathcal{A}$  in some round  $r \geq 0$  and  $\bar{n} = A \cup B$  a partition of  $\bar{n}$  such that*

$$(\forall b \in \Gamma_{\mathcal{A}}(3, 2))(b \subseteq A \text{ or } b \subseteq B). \quad (3)$$

*Then there exists the sequence  $\mathbf{p}: S \cdot \xi(A) \stackrel{B}{\sim} S \cdot \xi(\bar{3}) \stackrel{A}{\sim} S \cdot \xi(B)$  of connected states in round  $r + 1$  of  $\mathcal{A}$ .*

*Proof.* We show that the path  $\mathbf{p}$  can be build from the hypothesis. To do this, it is enough to show that there exists the possibility that the output values of the safe-consensus shared objects invoked by the processes are the same in the three states  $S \cdot \xi(A)$ ,  $S \cdot \xi(\bar{3})$  and  $S \cdot \xi(B)$ . We have cases, according to the way in which the processes invoke the safe-consensus objects in round  $r + 1$ .

*Case a).* If each process invokes in solo a safe-consensus object, then by the Safe-Validity property of safe-consensus, each process receives its own id as output value from the shared object it invokes, so that all processes see the same output values from the safe-consensus objects in the states  $S \cdot \xi(A)$ ,  $S \cdot \xi(\bar{3})$  and  $S \cdot \xi(B)$ .

*Case b).* Suppose that processes  $p_i, p_j$  invoke a shared object and  $p_k$  invokes in solo another safe-consensus object. This fact is represented by the 2-box  $b_1 = \{i, j\}$  and the trivial box  $b_2 = \{k\}$ . By the Safe-Validity property of safe-consensus,  $p_k$  always receives as output value from the shared object represented by  $b_2$  its own id, thus  $p_k$  sees the same safe-consensus value in the three states. Now, as  $b_1 \in \Gamma_{\mathcal{A}}(3, 2)$ , by Equation (3), we know that  $b_1 \subseteq A$  or  $b_1 \subseteq B$ , so that in each state of  $\mathbf{p}$ , processes  $p_i, p_j$  invoke concurrently the safe-consensus object represented by  $b_1$  and by the Safe-Validity property, the return value of the shared object can be arbitrary, thus there exists executions of  $\mathcal{A}$  in which we can make the safe-consensus object represented by  $b_1$  output the same value in the three states  $S \cdot \xi(A)$ ,  $S \cdot \xi(\bar{3})$  and  $S \cdot \xi(B)$ .

*Case c).* Now suppose that all three processes invoke the same safe-consensus object, which is represented by the 3-box  $b = \bar{3}$ . Because  $\bar{3} = A \cup B$  and  $A \cap B = \emptyset$ , it must be true that  $|b \cap A| = 2$  or  $|b \cap B| = 2$ . Without loss, assume that  $|b \cap A| = 2$ , then  $|b \cap B| = 1$  and by the Safe-Validity property, the output value of the shared object represented by  $b$  must be  $l \in b \cap B$  in the state  $S \cdot \xi(B)$  and in the states  $S \cdot \xi(A)$ ,  $S \cdot \xi(\bar{3})$ , the output value can be



arbitrary, this is because in these two states, at least two processes are invoking concurrently the safe-consensus object represented by  $b$ . Therefore there exists executions of  $\mathcal{A}$  in which the output value of the safe-consensus object is  $l$  in the three states  $S \cdot \xi(A)$ ,  $S \cdot \xi(\bar{3})$  and  $S \cdot \xi(B)$ . It follows that the path  $\mathbf{p}$  of the conclusion exists.  $\square$

**Lemma 4.5.** *Suppose that for the protocol  $\mathcal{A}$  there exist a partition  $\bar{n} = A \cup B$  satisfying Equation 3 and a sequence  $\mathbf{p}: S_0 \xrightarrow{X_1} \dots \xrightarrow{X_l} S_l$  of connected states in round  $r \geq 0$  of  $\mathcal{A}$ , such that  $X_i = A$  or  $X_i = B$  for all  $i \in \{1, \dots, l\}$ . Then in round  $r + 1$  of  $\mathcal{A}$  there exists a path  $\mathbf{q}: Q_0 \xrightarrow{Y_1} \dots \xrightarrow{Y_s} Q_s$  of connected states and the following properties hold:*

- I) Each state  $Q_k$  is of the form  $Q_k = S_j \cdot \xi(X)$ , where  $X = A$  or  $X = B$ ;
- II)  $(\forall j \in \{1, \dots, s\})(Y_j = A \text{ or } Y_j = B)$ .

*Proof.* To find the path  $\mathbf{q}$  satisfying I) and II), we use induction on  $l$ . In the base case  $l = 1$ ,  $\mathbf{p}: S_0 \xrightarrow{X_1} S_1$  with  $X_1 = A$  or  $X_1 = B$ . It is easy to see that the path  $S_0 \cdot \xi(X_1) \xrightarrow{X_1} S_1 \cdot \xi(X_1)$  fulfills conditions I), II). For the induction hypothesis, suppose that for the path  $S_0 \xrightarrow{X_1} \dots \xrightarrow{X_{l'}} S_{l'}$  with  $1 \leq l' < l$ , we have build the path  $\mathbf{q}': Q_1 \xrightarrow{Y_1} \dots \xrightarrow{Y_{s'}} Q_{s'}$  satisfying I) and II) of the conclusion of the Lemma. We now show how to connect  $Q_{s'}$  with a successor state of  $S_{l'+1}$ . Let  $X_{l'+1}$  be the set of processes that cannot distinguish between  $S_{l'}$  and  $S_{l'+1}$ . By the induction hypothesis,  $Q_{s'} = S_{l'} \cdot \xi(X)$ , where  $X = A$  or  $X = B$ . We have cases.

Case  $X = X_{l'+1}$ . In this case we use the small path  $S_{l'} \cdot \xi(X) \xrightarrow{X} S_{l'+1} \cdot \xi(X)$ .

Case  $X \neq X_{l'+1}$ . Without loss of generality, assume that  $X = A$  and  $X_{l'+1} = B$ . We apply Lemma 4.4 to obtain the path  $S_{l'} \cdot \xi(A) \xrightarrow{B} S_{l'} \cdot \xi(\bar{3}) \xrightarrow{A} S_{l'} \cdot \xi(B)$ . Combining this path with the path  $S_{l'} \cdot \xi(B) \xrightarrow{B} S_{l'+1} \cdot \xi(B)$ , we are done.

We have build with induction the sequence of connected states  $Q_1, \dots, Q_s$  from the sequence  $S_1, \dots, S_l$  satisfying the demanded properties. The result follows.  $\square$

**Lemma 4.6.** *If  $\mathcal{A}$  is a WOR protocol for three processes using safe-consensus objects with  $\nu_{\mathcal{A}}(3, 2) \leq 1$  and  $I$  is an initial state in  $\mathcal{A}$ , then there exists a partition of the set  $\bar{3} = A \cup B$  such that for all  $u \geq 1$ , the states  $I \cdot \xi^u(A)$  and  $I \cdot \xi^u(B)$  are connected.*

*Proof.* We can apply Lemma 4.3 to the set  $\Gamma_{\mathcal{A}}(3, 2) \subset V_{3,2}$  to find the partition of  $\bar{3}$  and then we use induction combined with Lemma 4.5. We omit the details.  $\square$

**Case  $m_0 = 3$**  The last case to consider is when the three processes cannot invoke the same safe-consensus shared object together. To prove this case, we need one structural result, regarding paths of connected states in an iterated protocol. With this result, we can build a bivalency argument to prove the lower bound for the case of  $m_0 = 3$  in the proof of Theorem 4.2.

**Lemma 4.7.** *Suppose that  $\mathcal{A}$  is a WOR protocol with safe-consensus objects for three processes such that  $\nu_{\mathcal{A}}(3, 3) = 0$ . If  $S, Q$  are two initial states in  $\mathcal{A}$ , connected with a sequence  $\mathbf{q}_0: S \sim \dots \sim Q$ , then for all  $u \geq 0$ , there exist successor states  $S^u, Q^u$  of  $S$  and  $Q$  respectively, in round  $u$  of  $\mathcal{A}$ , such that  $S^u$  and  $Q^u$  are connected.*

*Proof.* Let  $\mathcal{A}$  be a protocol for three processes with  $\nu_{\mathcal{A}}(3, 3) = 0$ . As for round  $u = 0$ , we have the path  $\mathbf{q}_0$ , to prove the result it is enough to show that if for  $u \geq 0$ , we have build the path  $\mathbf{q}_u: S^u \sim \dots \sim Q^u$ , connecting the states  $S^u$  and  $Q^u$ , then in round  $u + 1$  of  $\mathcal{A}$  we can find a new path

$$\mathbf{q}_{u+1}: S^{u+1} \sim \dots \sim Q^{u+1},$$

which connects  $S^{u+1}$  and  $Q^{u+1}$ , successor states of  $S^u$  and  $Q^u$  respectively.

Assume then that we have the path  $\mathbf{q}_u: S_0 \stackrel{X_1}{\sim} \dots \stackrel{X_q}{\sim} S_q$ , where  $S^u = S_1$  and  $S_q = Q^u$ . We proceed by induction on  $q$ . For the base case  $q = 1$ , we have that  $\mathbf{q}_u$  is the path  $S_0 \stackrel{X_1}{\sim} S_1$ , here we easily build the path  $S_0 \cdot \xi(X_1) \stackrel{X_1}{\sim} S_1 \cdot \xi(X_1)$ . Suppose that for  $1 \leq l < q$ , we have build the path  $\mathbf{q}': R_1 \sim \dots \sim R_s$ , where  $R_1$  is a successor state of  $S_1$  and  $R_s = S_l \cdot \xi(X)$  is a successor states of  $S_l$ . We now wish to connect  $R_s$  (a successor state of  $S_l$ ) with a successor state of  $S_{l+1}$ . Let  $X_{l+1}$  be the set of processes wich cannot distinguish between  $S_l$  and  $S_{l+1}$ . As  $\nu_{\mathcal{A}}(3, 3) = 0$ , In any execution of  $\mathcal{A}$ , the three processes cannot invoke the same safe-consensus shared object, thus in round  $u + 1$  of  $\mathcal{A}$ , they must invoke the safe-consensus objects in one of the following two possibilities:

- Each process invokes a safe-consensus object in solo.
- Two processes invoke a safe-consensus object and the third process invokes in solo another shared object.

If the processes invoke three separate safe-consensus objects, then we build the following path from  $R_s = S_l \cdot \xi(X)$  to  $S_{l+1} \cdot \xi(X_{l+1})$

$$S_l \cdot \xi(X) \stackrel{\bar{3}-X}{\sim} S_l \cdot \xi(\bar{3}) \stackrel{\bar{3}-X_{l+1}}{\sim} S_l \cdot \xi(X_{l+1}) \stackrel{X_{l+1}}{\sim} S_{l+1} \cdot \xi(X_{l+1}). \quad (4)$$

As each process sees its own id as the output value of the safe-consensus object it invokes, then the only way that a process can distinguish between two states, is by means of the contents of the shared memory. Therefore the path given in Equation (4) exists.

In case that two processes invoke a safe-consensus object, represented by the 2-box  $b \in \Gamma_{\mathcal{A}}(3, 2)$  and the other process invokes in solo an object represented by the trivial box  $c$ , then we build a path from  $R_s = S_l \cdot \xi(X)$  to  $S_{l+1} \cdot \xi(X_{l+1})$  as follows: First notice that the two states  $S_l \cdot \xi(X)$  and  $S_l \cdot \xi(\bar{3})$  are indistinguishable for the processes with ids in the set  $\bar{3} - X$ , this is because we can find executions of  $\mathcal{A}$  in which the safe-consensus values of the objects represented by  $b$  and  $c$  are the same in the two previous states (Safe-Validity), proving this is an easy case analysis. Now, we need to connect the state  $S_l \cdot \xi(\bar{3})$  with the state  $S_l \cdot \xi(X_{l+1})$ . We have subcases on the size of the set  $X_{l+1}$ .

Case  $|X_{l+1}| = 1$ . If  $X_{l+1} = c$ , then we have that  $S_l \cdot \xi(\bar{3}) \stackrel{b}{\sim} S_l \cdot \xi(X_{l+1})$ , because by the Safe-Validity property of safe-consensus, we can find executions of  $\mathcal{A}$  in which the output value of the safe-consensus object represented by  $b$  is the same in the states  $S_l \cdot \xi(\bar{3})$  and  $S_l \cdot \xi(X_{l+1})$ , thus all processes with ids in  $b$  cannot distinguish between these two states. On the other hand, if  $X_{l+1} \neq c$ , then  $X_{l+1} \subset b$  and we have the path  $S_l \cdot \xi(\bar{3}) \stackrel{c}{\sim} S_l \cdot \xi(X_{l+1})$ .

Case  $|X_{l+1}| = 2$ . If  $X_{l+1} = b$ , then we claim that, as in the last part of the previous case,  $S_l \cdot \xi(\bar{3}) \stackrel{c}{\sim} S_l \cdot \xi(X_{l+1})$ . When  $X_{l+1} \neq b$ , it must be true that  $X_{l+1} = \{j\} \cup c$ , where  $j \in b$ . The path that we need to build here is

$$S_l \cdot \xi(\bar{3}) \stackrel{c}{\sim} S_l \cdot \xi(\{j\}) \stackrel{b}{\sim} S_l \cdot \xi(\{j\}, c) \stackrel{c}{\sim} S_l \cdot \xi(\{j\} \cup c).$$

The arguments to prove that this path exists, are very similar to previous arguments, using the Safe-Validity property of the safe-consensus task. This finishes the cases to connect the states  $S_l \cdot \xi(\bar{3})$  and  $S_l \cdot \xi(X_{l+1})$ .

Finally, we connect the states  $S_l \cdot \xi(X_{l+1})$  and  $S_{l+1} \cdot \xi(X_{l+1})$  with the small path  $S_l \cdot \xi(X_{l+1}) \stackrel{X_{l+1}}{\sim} S_{l+1} \cdot \xi(X_{l+1})$ . Thus, we have connected the state  $R_s = S_l \cdot \xi(X)$  with a successor state of  $S_{l+1}$  and this completes the proof of the induction step.

Therefore we have proven using induction, that given the path  $\mathbf{q}_u$ , connecting the states  $S_1 = S^u$  and  $S_q = Q^u$ , we can build a new path  $\mathbf{q}_{u+1}$ , connecting successor states of  $S_1 = S^u$  and  $S_q = Q^u$  respectively. This finishes the proof.  $\square$

**Proof of Theorem 4.2** (*Case  $n = 3$* ) Assume that there is some  $m_0 \in \{2, 3\}$  such that  $\nu_{\mathcal{A}}(3, m_0) \leq 3 - m_0$ . Let  $O, U$  be the initial states in which all processes have as input values 0s and 1s respectively. We now find successor states of  $O$  and  $U$  in each round  $r \geq 0$ , which are connected. We have cases:

*Case  $m_0 = 2$ .* By Lemma 4.6, there exists a partition of  $\bar{3} = A \cup B$  such that for any state  $S$  and any  $r \geq 0$ ,  $S \cdot \xi^r(A)$  and  $S \cdot \xi^r(B)$  are connected. Let  $OU$  be the initial state in which all processes with ids in  $A$  have as input value 0s and all processes with ids in  $B$  have as input values 1s. Then for all  $r \geq 0$

$$O \cdot \xi^r(A) \stackrel{A}{\sim} OU \cdot \xi^r(A) \quad \text{and} \quad OU \cdot \xi^r(B) \stackrel{B}{\sim} U \cdot \xi^r(B)$$

and by Lemma 4.6, the states  $OU \cdot \xi^r(A)$  and  $OU \cdot \xi^r(B)$  are connected. Thus, for any  $r$ , we can connect the states  $O^r = O \cdot \xi^r(A)$  and  $U^r = U \cdot \xi^r(B)$ .

*Case  $m_0 = 3$ .* By Lemma 3.2, we know that any two initial states for consensus are connected, so that we can connect  $O$  and  $U$  with a sequence  $\mathbf{q}$  of initial states of  $\mathcal{A}$ . By Lemma 4.7, for each round  $r \geq 0$  of  $\mathcal{A}$ , there exist successor states  $O^r, U^r$  of  $O$  and  $U$  respectively, such that  $O^r$  and  $U^r$  are connected.

In this way, we have connected successor states of  $O$  and  $U$  in each round of the protocol  $\mathcal{A}$ . Now,  $O$  is a 0-valent, initial state, which is connected to the initial state  $U$ , so that we can apply Lemma 3.3 to conclude that  $U$  is 0-valent. But this contradicts the fact that  $U$  is a 1-valent state, so we have reached a contradiction. Therefore  $\nu_{\mathcal{A}}(3, m) > 3 - m$  for  $m = 2, 3$ .

With the  $n = 3$  version of Theorem 4.2 at hand, We have the following equalities, proving the lower bound result for the case of  $n = 3$  processes.

$$\nu_{\mathcal{A}}(3) = \nu_{\mathcal{A}}(3, 2) + \nu_{\mathcal{A}}(3, 3) \geq (3 - 2 + 1) + (3 - 3 + 1) = 3 = \binom{3}{2}.$$

## 5 The impossibility of consensus in the WRO iterated model

In this section, we give a negative result concerning protocols in the WRO iterated model; namely, we show that the following question

*Is every task solvable in read/write shared memory with the addition of safe-consensus objects, solvable by a protocol in the WRO iterated model ?*

It does not have a positive answer. Although there exists wait-free shared memory protocols that can solve consensus using safe-consensus objects [2], in this paper we show that there is no protocol in the WRO iterated model that solves consensus using safe-consensus objects. For simplicity, we will refer to a protocol in the WRO iterated model simply as a WRO protocol.

### 5.1 The connectivity of protocols in the WRO iterated model

In order to prove the impossibility of consensus in the WRO iterated model (Theorem 5.4), we need various results that describe the structure of protocols in the WRO iterated model. These results tell us that for any given WRO protocol, the connectivity between some specific reachable states is high, even with the added power of safe-consensus objects. This is the main reason of why consensus cannot be implemented with safe-consensus in the WRO iterated model.

**Lemma 5.1.** *Let  $\mathcal{A}$  be a WRO protocol for  $n$  processes,  $A_i = \bar{n} - i$  for some  $i \in \bar{n}$  and  $S, Q$  two reachable states of  $\mathcal{A}$  in round  $r$ , such that for all  $j \in A_i$ ,  $p_j$  has the same snapshot information in  $S$  and  $Q$ . Then  $\mathbf{Inv}(S) = \mathbf{Inv}(Q)$ .*

*Proof.* Let  $\mathcal{A}$  be a WRO protocol,  $A_i = \bar{n} - i$  and  $S, Q$  such that every process  $p_j$  with  $j \in A_i$  has the same snapshot information in  $S$  and  $Q$ . By Lemma 3.1, if  $b \in \mathbf{Inv}(S)$  is a box such that  $i \notin b$ , then  $b \in \mathbf{Inv}(Q)$ . For the box  $c \in \mathbf{Inv}(S)$  that contains the id  $i$ , we argue by cases:

*Case I.*  $|c| = 1$ . All processes with ids in  $A_i$  participate in the same boxes in both states  $S$  and  $Q$  and  $p_i$  does not participate in those boxes, thus  $c = \{i\} \in \mathbf{Inv}(Q)$ .

*Case II.*  $|c| > 1$ . There exists a process  $p_j$  with  $j \in c$  and  $j \neq i$ . Then  $j \in A_i$ , so that by Lemma 3.1,  $c \in \mathbf{Inv}(Q)$ .

We have proven that  $\mathbf{Inv}(S) \subseteq \mathbf{Inv}(Q)$  and to show that the other inclusion holds, the argument is symmetric. Therefore  $\mathbf{Inv}(S) = \mathbf{Inv}(Q)$ .  $\square$

Our next result describes some of the structure of WRO protocols, it will be most helpful to show Theorem 5.4. The following construction will be useful for the rest of the section. If  $A_1, \dots, A_q \subset \bar{n}$  ( $q \geq 1$ ) is a collection of disjoint subsets of  $\bar{n}$ , define the round schedule  $\eta(A_1, \dots, A_q, Y)$  for  $\mathcal{A}$  as:

$$W(A_1), R(A_1), W(A_2), R(A_2), \dots, W(A_q), R(A_q), W(Y), R(Y), S(\bar{n})$$

where  $Y = \bar{n} - (\bigcup_{i=1}^q A_i)$ . Sometimes, we omit the set  $Y$  and just write  $\eta(A_1, \dots, A_q)$ .

**Lemma 5.2.** *Let  $n \geq 2$ ,  $\mathcal{A}$  a WRO protocol with safe-consensus objects and  $i, j \in \bar{n}$ . Then for any reachable state  $S$  in round  $r \geq 0$  of  $\mathcal{A}$ , the states  $S \cdot \eta(\bar{n} - i)$  and  $S \cdot \eta(\bar{n} - j)$  are connected in round  $r + 1$  of  $\mathcal{A}$  with a C-regular path  $\mathbf{q}$  with  $\deg \mathbf{q} \geq n - 1$ .*

*Proof.* If  $i = j$ , the result is immediate. Suppose that  $i \neq j$  and  $\bar{n} - i = \{l_1, \dots, l_{n-1}\}$  with  $l_{n-1} = j$ . We show that  $Q_i = S \cdot \eta(\bar{n} - i)$  and  $Q_j = S \cdot \eta(\bar{n} - j)$  can be connected with a C-regular sequence with indistinguishability degree  $n - 1$  by building three subsequences  $\mathbf{q}_1, \mathbf{q}_2$  and  $\mathbf{q}_3$ . Using Lemma 5.1 and the properties of the safe-consensus task to find executions of  $\mathcal{A}$  in which the safe-consensus values are adequate, we can show that the first path  $\mathbf{q}_1$  is given by

$$S \cdot \eta(\bar{n} - i, i) \stackrel{\bar{n}-l_1}{\sim} S \cdot \eta(l_1, \bar{n} - \{i, l_1\}, i) \stackrel{\bar{n}-l_2}{\sim} \dots \stackrel{\bar{n}-l_{n-2}}{\sim} S \cdot \eta(l_1, \dots, l_{n-1}, i),$$

which is clearly a C-regular path by Lemma 5.1. In the same way, we construct the sequence  $\mathbf{q}_2$  as

$$S \cdot \eta(l_1, \dots, l_{n-1}, i) \stackrel{\bar{n}-l_{n-1}}{\sim} S \cdot \eta(l_1, \dots, \{l_{n-1}, i\}) \stackrel{\bar{n}-i}{\sim} S \cdot \eta(l_1, \dots, i, l_{n-1})$$

and finally, in a very similar way, we build  $\mathbf{q}_3$  as follows,

$$\begin{aligned} S \cdot \eta(l_1, \dots, i, l_{n-1}) &\stackrel{\bar{n}-l_{n-2}}{\sim} S \cdot \eta(l_1, \dots, \{l_{n-2}, i\}, l_{n-1}) \\ &\stackrel{\bar{n}-l_{n-3}}{\sim} \dots \stackrel{\bar{n}-l_2}{\sim} \\ &S \cdot \eta(l_1, \bar{n} - \{l_1, l_{n-1}\}, l_{n-1}) \stackrel{\bar{n}-l_1}{\sim} S \cdot \eta(\bar{n} - l_{n-1}, l_{n-1}). \end{aligned}$$

Applying Lemma 5.1, both  $\mathbf{q}_2$  and  $\mathbf{q}_3$  are C-regular paths and for  $k = 1, 2, 3$ ,  $\deg \mathbf{q}_k = n - 1$ . Notice that the processes execute the safe-consensus objects in the same way in all the states of the previous sequences, and by Lemma 5.1, all the states have the same invocation specification. With all this data, we can find executions of  $\mathcal{A}$  in which the states of the sequences  $\mathbf{q}_s$  ( $s = 1, 2, 3$ ) have the same safe-consensus value for all the boxes. As  $j = l_{n-1}$ ,  $S \cdot \eta(\bar{n} - l_{n-1}, l_{n-1}) = S \cdot \eta(\bar{n} - j, j) = Q_j$ , thus we can join  $\mathbf{q}_1, \mathbf{q}_2$  and  $\mathbf{q}_3$  to obtain a C-regular path  $\mathbf{q}$ :  $Q_i \sim \dots \sim Q_j$  such that  $\deg \mathbf{q} = n - 1$ . This finishes the proof.  $\square$

**Lemma 5.3.** *Let  $\mathcal{A}$  be a WRO protocol and  $S^r, P^r$  be two reachable states of some round  $r \geq 0$  such that  $S^r$  and  $P^r$  are connected with a sequence  $\mathbf{s}$  such that  $\deg \mathbf{s} \geq n - 1$ ; suppose also that the number of participating processes is  $n \geq 2$ . Then there exists successor states  $S^{r+1}, P^{r+1}$  of  $S^r$  and  $P^r$  respectively, in round  $r + 1$  of  $\mathcal{A}$ , such that  $S^{r+1}$  and  $P^{r+1}$  are connected with a C-regular path  $\mathbf{q}$  with  $\deg \mathbf{q} \geq n - 1$ .*

*Proof.* Suppose that  $S^r$  and  $P^r$  are connected with a sequence  $\mathbf{s}$ :  $S^r = S_1 \sim \dots \sim S_m = P^r$  such that for all  $j$  ( $1 \leq j \leq m - 1$ ),  $S_j \stackrel{X}{\sim} S_{j+1}$ , where  $|X| \geq n - 1$ . Let  $p_1, \dots, p_n$  be the set of participating processes. We now construct a sequence of connected states

$$\mathbf{q}: Q_1 \sim \dots \sim Q_s,$$

in such a way that each  $Q_i$  is a successor state of some  $S_j$  and  $\deg \mathbf{q} \geq n - 1$ . We use induction on  $m$ ; for the basis, consider the adjacent states  $S_1, S_2$  and suppose that all processes with ids in the set  $X_1$  ( $|X_1| \geq n - 1$ ) cannot distinguish between them. By Lemma 5.1, the states  $Q_1 = S_1 \cdot \eta(X_1)$  and  $Q_2 = S_2 \cdot \eta(X_1)$  satisfy the equality  $\mathbf{Inv}(Q_1) = \mathbf{Inv}(Q_2)$  and with appropriate executions of  $\mathcal{A}$ , we can see that  $Q_1$  and  $Q_2$  are indistinguishable for  $X_1$ , and so the small sequence  $Q_1 \stackrel{X_1}{\sim} Q_2$  has indistinguishability degree at least  $n - 1$ . Assume now that we have build the sequence

$$\mathbf{q}': Q_1 \sim \dots \sim Q_{s'},$$

of connected successor states for  $S_1 \sim \dots \sim S_q$  ( $1 \leq q < m$ ) such that  $\mathbf{q}'$  is C-regular,  $\deg \mathbf{q}' \geq n - 1$  and  $Q_{s'} = S_q \cdot \eta(X)$  with  $|X| \geq n - 1$ . Let  $X_q$  with  $|X_q| \geq n - 1$  be a set of processes' ids that cannot distinguish between  $S_q$  and  $S_{q+1}$ . To connect  $Q_{s'}$  with a successor state for  $S_{q+1}$ , we first use Lemma 5.2 to connect  $Q_{s'}$  and  $Q = S_q \cdot \eta(X_q)$  by means of a C-regular sequence  $\mathbf{p}$  such that  $\deg \mathbf{p} \geq n - 1$ . Second, notice that we have the small C-regular path  $\mathbf{s}: Q \stackrel{X_q}{\sim} S_{q+1} \cdot \eta(X_q) = Q_{s'+1}$ . In the end, we use  $\mathbf{q}', \mathbf{p}$  and  $\mathbf{s}$  to get a C-regular sequence

$$\mathbf{q}: Q_1 \sim \dots \sim Q_{s'+1},$$

which fulfills the inequality  $\deg \mathbf{q} \geq n - 1$ . By induction, the result follows.  $\square$

**Theorem 5.4.** *There is no protocol for consensus in the WRO iterated model using safe-consensus objects.*

*Proof.* Suppose  $\mathcal{A}$  is a WRO protocol that solves consensus with safe-consensus objects and without loss, assume that 0,1 are two valid input values. Let  $O, U$  be the initial states in which all processes have as initial values only 0 and 1 respectively. By Lemma 3.2,  $O$  and  $U$  are connected and it is easy to see that the sequence joining  $O$  and  $U$ , build in the proof on Lemma 3.2 has degree of indistinguishability  $n - 1$ . So that applying an easy induction, we can use Lemma 5.3 to show that there exists connected successor states  $O^r, U^r$  of  $O$  and  $U$  respectively in each round  $r \geq 0$  of the protocol  $\mathcal{A}$ . Clearly,  $O$  is a 0-valent state and by Lemma 3.3,  $U$  is 0-valent. But this is a contradiction, because  $U$  is a 1-valent state. Therefore no such protocol  $\mathcal{A}$  can exists.  $\square$

## 6 The equivalence of the WRO and OWR iterated models

In this section, we prove Theorem 6.4, which tell us that the WRO and the OWR iterated models have the same computational power. To check the meaning of some definition or previous result, the reader may consult Sections 2, 3.3 and 3.4. For the sake of simplicity, we will refer to a protocol in the OWR iterated model, simply as a OWR protocol.

### 6.1 Transforming a WRO protocol into an OWR protocol

The algorithm of Figure 6 is a generic OWR protocol to simulate protocols in the WRO iterated model. Suppose that  $\mathcal{A}$  is a WRO protocol that solves a task  $\Delta$  and assume that  $h_{\mathcal{A}}$  is the deterministic function to select safe-consensus objects in  $\mathcal{A}$  and  $\delta_{\mathcal{A}}$  is the decision map used in the protocol  $\mathcal{A}$ . To obtain a OWR protocol  $\mathcal{B}$  that simulates the behavior of  $\mathcal{A}$ , we use the generic protocol of Figure 6, replacing the functions  $h_{\perp}$  and  $\delta_{\perp}$  with  $h_{\mathcal{A}}$  and  $\delta_{\mathcal{A}}$  respectively. If the processes execute  $\mathcal{B}$  with valid input values from  $\Delta$ , then in the first round, the participating processes discard the output value of the safe-consensus object that they invoke, because the test at line (6) is successful and after that, each process goes on to execute the write-snapshot operations, with the same values that they would use to perform the same operations in round one of  $\mathcal{A}$ ; later, at lines (13)-(15), they do some local computing to try to decide an output value. It is clear from the code of the generic protocol that none of the executing processes will write a non- $\perp$  value to the local variables  $dec$ , thus no process makes a decision in the first round and they finish the current round, only with two simulated operations from  $\mathcal{A}$ : write and snapshot. In the second round, the processes invoke one or more safe-consensus objects at line (5), accordingly to the return values of the function  $h_{\mathcal{A}}$ , which is invoked with the values obtained from the snapshot operation of the previous round (from the round-one write-read simulation of  $\mathcal{A}$ ) and notice that instead of using the current value of the round counter  $r$ , the processes call  $h_{\mathcal{A}}$  with the parameter  $r - 1$ . As  $r = 2$ , then the processes invoke  $h_{\mathcal{A}}$  as if they were still executing the first round of the simulated protocol  $\mathcal{A}$ . Finally, after using the safe-consensus objects, processes do local computations in lines (8)-(10), using the decision map  $\delta_{\mathcal{A}}$  to simulate the decision phase of  $\mathcal{A}$ , if for some process  $p_j$ , it is time to take a decision in  $\mathcal{A}$ , it stores the output value in the local variable  $dec'_j$ , which is used at the end of the round to write the output value in  $dec_j$  and then  $p_j$  has decided. If the map  $\delta_{\mathcal{A}}$  returns  $\perp$ , then  $p_j$  goes on to do the next write-read operations (simulating the beginning of round two of  $\mathcal{A}$ ) and proceeds to round three of  $\mathcal{B}$ . The described behavior is going to repeat in all subsequent rounds.

**Algorithm WRO-Simulation**

```

(1) init  $r \leftarrow 0$ ;  $sm, tmp \leftarrow input$ ;  $dec, dec' \leftarrow \perp$ ;  $val \leftarrow \perp$ ;

(2) loop forever
(3)    $r \leftarrow r + 1$ ;
(4)    $sm \leftarrow tmp$ ;
(5)    $val \leftarrow S[h_-(r - 1, id, sm, val)].exec(id)$ ;
(6)   if ( $r = 1$ ) then
(7)      $val \leftarrow \perp$ ; /* 1st round: ignore returned value */
(8)   else if ( $dec' = \perp$ ) then
(9)      $dec' \leftarrow \delta_-(sm, val)$ ; /* Decision of simulated protocol */
(10)  end if
(11)   $SM[r].update(sm, val)$ ;
(12)   $tmp \leftarrow SM[r].scan()$ ;
(13)  if ( $dec = \perp$ ) then
(14)     $dec \leftarrow dec'$ ; /* Decide when simulated protocol decides */
(15)  end if
(16) end loop

```

Figure 6: General WRO-simulation protocol in the OWR iterated model

### Correspondence between executions of the protocols

Let  $\pi$  be a round schedule. Denote by  $\pi[W, R]$  the sequence of write and read events of  $\pi$ , these events must appear in  $\pi[W, R]$  just in the same order they are specified in  $\pi$ . The symbol  $\pi[S]$  is defined for the event  $S$  in a similar way. For example, given the round schedule

$$\pi = W(1, 3), R(1, 3), W(2), R(2), S(1, 2, 3),$$

then  $\pi[W, R] = W(1, 3), R(1, 3), W(2), R(2)$  and  $\pi[S] = S(1, 2, 3)$ . If

$$\pi' = W(1, 2, 3), R(1, 2, 3), S(1), S(3), S(2),$$

we have that  $\pi[W, R] = W(1, 2, 3), R(1, 2, 3)$  and  $\pi[S] = S(1), S(3), S(2)$ .

Let  $\mathcal{A}$  be a WRO protocol as given in Figure 2,  $h_{\mathcal{A}}$  the deterministic function to select safe-consensus objects and  $\delta_{\mathcal{A}}$  the decision map used in the protocol  $\mathcal{A}$ . A OWR protocol  $\mathcal{B}$  that simulates the behavior of  $\mathcal{A}$  is obtained by using the generic protocol  $\mathcal{G}$  of Figure 6. To construct  $\mathcal{B}$ , we only replace the functions  $h$  and  $\delta'$  of  $\mathcal{G}$  with  $h_{\mathcal{A}}$  and  $\delta_{\mathcal{A}}$  respectively.

In order to show that  $\mathcal{B}$  simulates  $\mathcal{A}$ , we first notice that there is a correspondence between executions of  $\mathcal{A}$  and  $\mathcal{B}$ . For, if  $\alpha$  is an execution of  $\mathcal{A}$  such that

$$\alpha = S_0, \pi_1, S_1, \dots, \pi_k, S_k, \pi_{k+1}, \dots$$

Then there exists an execution  $\alpha_{\mathcal{B}}$  of  $\mathcal{B}$

$$\alpha_{\mathcal{B}} = S_0, \pi'_1, S'_1, \dots, \pi'_k, S'_k, \pi'_{k+1}, \dots,$$

such that

$$\pi'_l = \begin{cases} S(X_1) \dots, S(X_s), \pi_1[W, R] & \text{if } l = 1, \\ \pi_{l-1}[S], \pi_l[W, R] & \text{otherwise,} \end{cases}$$

where  $\bigcup_{j=1}^s X_j = \bar{n}$  is an arbitrary partition of  $\bar{n}$ . Conversely, for any execution  $\beta = R_0, \eta_1, R_1, \dots, \eta_k, R_k, \eta_{k+1}, \dots$  of the protocol  $\mathcal{B}$ , we can find an execution

$$\beta_{\mathcal{A}} = R_0, \eta'_1, R'_1, \dots, \eta'_k, R'_k, \eta_{k+1}, \dots$$

of  $\mathcal{A}$  such that

$$\eta'_l = \eta_l [W, R], \eta_{l+1} [S] \quad \text{for all } l \geq 1.$$

**Lemma 6.1.** *Suppose that  $\beta$  is an execution of the protocol  $\mathcal{B}$  and process  $p_i$  is executing the protocol (accordingly to  $\beta$ ) in round  $r > 0$  at line 11. Then the local variables  $sm_i, dec'_i$  and  $val_i$  of  $\mathcal{B}$  have the same values of the respectively local variables  $sm_i, dec_i$  and  $val_i$  of the protocol  $\mathcal{A}$ , when  $p_i$  finishes executing round  $r - 1$  of  $\mathcal{A}$ , in the way specified by the execution  $\beta_{\mathcal{A}}$ .*

*Proof.* The proof is based on induction on the round number  $r$ , the base case and the inductive case are obtained by an easy analysis of the programs given in Figures 2 and 6.  $\square$

The converse of the previous lemma is also true.

**Lemma 6.2.** *Suppose that  $\alpha$  is an execution of the protocol  $\mathcal{A}$  and process  $p_i$  is executing the protocol (accordingly to  $\alpha$ ) in round  $r > 0$  at line 4. Then the local variables  $sm_i, dec_i$  and  $val_i$  of  $\mathcal{A}$  have the same values of the respectively local variables  $sm_i, dec'_i$  and  $val_i$  of the protocol  $\mathcal{B}$ , when  $p_i$  finishes executing round  $r + 1$  of  $\mathcal{B}$ , in the way specified by the execution  $\alpha_{\mathcal{B}}$ .*

The final result that we need to prove that  $\mathcal{B}$  simulates  $\mathcal{A}$  for task solvability is an immediate consequence of Lemma 6.1.

**Lemma 6.3.** *Suppose that  $\Delta$  is a decision task solved by the protocol  $\mathcal{A}$  and let  $p_i \in \Pi$ . Then  $p_i$  decides an output value  $v$  in round  $r > 0$  of an execution  $\alpha$  of  $\mathcal{A}$  if and only if  $p_i$  decides the output value  $v$  in round  $r + 1$  of  $\mathcal{B}$ , in the execution  $\alpha_{\mathcal{B}}$ .*

This shows that the protocol  $\mathcal{B}$  simulates the behavior of  $\mathcal{A}$  for task solvability and therefore proves the first part of Theorem 6.4.

## 6.2 Transforming an OWR protocol into a WRO protocol

Now we show that any OWR protocol can be simulated with a WRO protocol. We follow the same technique that we used in the previous section, we use the generic protocol of Figure 7. The intuitive argument of how it works is very much the same as in the WRO case, thus we omit the details.

**Theorem 6.4.** *Let  $\mathcal{A}$  be a protocol in the WRO iterated model which solves a task  $\Delta$ . Then  $\mathcal{A}$  can be simulated with a protocol  $\mathcal{B}$  in the OWR iterated model. Conversely, for every protocol  $\mathcal{B}'$  in the OWR iterated model that solves a task  $\Delta'$ , there is a protocol  $\mathcal{A}'$  in the WRO iterated model, which simulates  $\mathcal{B}'$ .*

The equivalence of the WRO iterated model with the OWR iterated model can be combined with Theorem 5.4 to obtain the following

**Corollary 6.5.** *There is no protocol in the OWR iterated model for consensus using safe-consensus objects.*



```

Algorithm OWR-Simulation
(1) init  $r \leftarrow 0$ ;  $sm \leftarrow input$ ;  $dec, dec' \leftarrow \perp$ ;  $val \leftarrow \perp$ ;

(2) loop forever
(3)    $r \leftarrow r + 1$ ;
(4)    $SM[r].update(sm, val)$ ;
(5)    $sm \leftarrow SM[r].scan()$ ;
(6)   if  $(r = 1)$  then
(7)      $sm \leftarrow input$ ;      /* 1st round: ignore write-snapshot */
(8)   else if  $(dec' = \perp)$  then
(9)      $dec' \leftarrow \delta_-(sm, val)$ ;
(10)  end if
(11)   $val \leftarrow S[h_-(r, id, sm, val)].exec(id)$ ;
(12)  if  $(dec = \perp)$  then
(13)     $dec \leftarrow dec'$ ;
(14)  end if
(15) end loop

```

Figure 7: General OWR-simulation protocol in the WRO iterated model

## 7 Conclusion

In this paper, we have introduced three extensions of the basic iterated model of distributed computing [12], using the safe-consensus task proposed by Yehuda, Gafni and Lieber in [2]. In the first iterated model, the WOR iterated model, processes write to memory, invoke safe-consensus objects and finally they snapshot the shared memory. We first constructed a WOR protocol which can solve  $n$ -consensus with  $\binom{n}{2}$  safe-consensus objects. To make this protocol more readable, we introduced a new group consensus task: The  $g$ -2coalitions-consensus task. We also proved that our WOR consensus protocol is sharp, by giving a  $\binom{n}{2}$  lower bound on the number of safe-consensus objects necessary to implement  $n$ -consensus in the WOR iterated model with safe-consensus. This lower bound is the main result of this paper, the full proof is somewhat complicated. At the very end, it is bivalency [22], but in order to be able to use bivalency, an intricate connectivity argument must be developed, in which we relate the way that the processes use the safe-consensus shared objects with subgraphs of the Johnson graphs. As connectivity of graphs appear again in the scene for consensus, this suggest that topology will play an important role to understand better the behavior of protocols that use shared objects more powerful than read/write shared memory registers. The proofs developed to obtain the lower bound say that the connectivity of the topological structures given by the shared objects used by the processes, can affect the connectivity of the topology of the protocol complex [28].

For the second iterated model that we investigated, the WRO iterated model, the processes first write to memory, then they snapshot it and after that, they invoke safe-consensus objects. We proved that in this model, the consensus task cannot be implemented. The impossibility proof uses simpler connectivity arguments than those used in the lower bound proof of the WOR iterated model. Finally, in the third iterated model, the OWR iterated model, processes first invoke safe-consensus objects, then they write to memory and then they snapshot the contents of the shared memory. We proved that this model is equivalent to the WRO iterated model for task solvability, thus we obtained as a corollary that consensus cannot be implemented in the OWR iterated model.

Concerning the relationship of the standard model of distributed computing [27, 32] extended with safe-consensus objects used in [2] to implement consensus using safe-consensus objects, we can conclude the following. Our results for the WRO and the OWR iterated models say that these two models are not equivalent to the standard model with safe-consensus (for task solvability), as consensus can be implemented in the latter model [2], but consensus cannot be implemented in the WRO and OWR iterated models. But an open problem related to these two iterated models is the following: Characterize their exact computational power.

On the other hand, the relationship between the standard model extended with safe-consensus and the WOR iterated model remains unknown. Are these two models equivalent? We conjecture that the answer is yes.

## References

- [1] Yehuda Afek, Hagit Attiya, Danny Dolev, Eli Gafni, Michael Merritt, and Nir Shavit. Atomic snapshots of shared memory. *J. ACM*, 40(4):873–890, 1993.
- [2] Yehuda Afek, Eli Gafni, and Opher Lieber. Tight group renaming on groups of size  $g$  is equivalent to  $g$ -consensus. In *Proceedings of the 23rd international conference on Distributed computing (DISC’09)*, volume 5805 of *LNCS*, pages 111–126, Berlin, Heidelberg, 2009. Springer-Verlag.
- [3] Yehuda Afek, Iftah Gamzu, Irit Levy, Michael Merritt, and Gadi Taubenfeld. Group renaming. In *OPODIS ’08: Proceedings of the 12th International Conference on Principles of Distributed Systems*, volume 5401 of *LNCS*, pages 58–72, Berlin, Heidelberg, 2008. Springer-Verlag.
- [4] Hagit Attiya, Amotz Bar-Noy, Danny Dolev, David Peleg, and Rudiger Reischuk. Renaming in an Asynchronous Environment. *Journal of the ACM*, July 1990.
- [5] Hagit Attiya, Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Bounds on the time to reach agreement in the presence of timing uncertainty. *J. ACM*, 41:122–152, January 1994.
- [6] Hagit Attiya and Sergio Rajsbaum. The combinatorial structure of wait-free solvable tasks. *SIAM J. Comput.*, 31(4):1286–1313, 2002.
- [7] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, 2004.
- [8] B. Bollobás. *Modern Graph Theory: Béla Bollobás*. Graduate Texts in Mathematics Series. Springer, 1998.
- [9] E. Borowsky, E. Gafni, N. Lynch, and S. Rajsbaum. The BG distributed simulation algorithm. *Distrib. Comput.*, 14(3):127–146, 2001.
- [10] Elizabeth Borowsky and Eli Gafni. Generalized FLP impossibility result for  $t$ -resilient asynchronous computations. In *STOC ’93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 91–100, New York, NY, USA, 1993. ACM.
- [11] Elizabeth Borowsky and Eli Gafni. Immediate atomic snapshots and fast renaming. In *PODC ’93: Proceedings of the twelfth annual ACM symposium on Principles of distributed computing*, pages 41–51, New York, NY, USA, 1993. ACM.

- [12] Elizabeth Borowsky and Eli Gafni. A simple algorithmically reasoned characterization of wait-free computation (extended abstract). In *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 189–198, New York, NY, USA, 1997. ACM.
- [13] Armando Castañeda, Maurice Herlihy, and Sergio Rajsbaum. An equivariance theorem with applications to renaming. In *Proceedings of the 10th Latin American International Conference on Theoretical Informatics, LATIN'12*, pages 133–144, Berlin, Heidelberg, 2012. Springer-Verlag.
- [14] Armando Castañeda, Damien Imbs, Sergio Rajsbaum, and Michel Raynal. Renaming is weaker than set agreement but for perfect renaming: A map of sub-consensus tasks. In *Proceedings of the 10th Latin American International Conference on Theoretical Informatics, LATIN'12*, pages 145–156, Berlin, Heidelberg, 2012. Springer-Verlag.
- [15] Armando Castañeda and Sergio Rajsbaum. New combinatorial topology upper and lower bounds for renaming. In *Proceedings of the Twenty-seventh ACM Symposium on Principles of Distributed Computing, PODC '08*, pages 295–304, New York, NY, USA, 2008. ACM.
- [16] Armando Castañeda and Sergio Rajsbaum. New combinatorial topology bounds for renaming: The upper bound. *J. ACM*, 59(1):3:1–3:49, March 2012.
- [17] Armando Castañeda, Sergio Rajsbaum, and Michel Raynal. The renaming problem in shared memory systems: An introduction. *Computer Science Review*, 5(3):229–251, 2011.
- [18] Soma Chaudhuri. More choices allow more faults: set consensus problems in totally asynchronous systems. *Inf. Comput.*, 105:132–158, July 1993.
- [19] Rodolfo Conde and Sergio Rajsbaum. An introduction to the topological theory of distributed computing with safe-consensus. *Electronic Notes in Theoretical Computer Science*, 283(0):29 – 51, 2012. Proceedings of the workshop on Geometric and Topological Methods in Computer Science (GETCO).
- [20] Rodolfo Conde and Sergio Rajsbaum. The complexity gap between consensus and safe-consensus. In Magnús M. Halldórsson, editor, *Structural Information and Communication Complexity*, volume 8576 of *Lecture Notes in Computer Science*, pages 68–82. Springer International Publishing, 2014.
- [21] Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a byzantine environment: Crash failures. *Information and Computation*, 88(2):156 – 186, 1990.
- [22] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
- [23] Eli Gafni. The extended BG-simulation and the characterization of t-resiliency. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 85–92, New York, NY, USA, 2009. ACM.

- [24] Eli Gafni and Sergio Rajsbaum. Distributed programming with tasks. In Chenyang Lu, Toshimitsu Masuzawa, and Mohamed Mosbah, editors, *Principles of Distributed Systems*, volume 6490 of *Lecture Notes in Computer Science*, pages 205–218. Springer Berlin Heidelberg, 2010.
- [25] Eli Gafni and Sergio Rajsbaum. Recursion in distributed computing. In Shlomi Dolev, Jorge Cobb, Michael Fischer, and Moti Yung, editors, *Stabilization, Safety, and Security of Distributed Systems*, volume 6366 of *Lecture Notes in Computer Science*, pages 362–376. Springer Berlin Heidelberg, 2010.
- [26] Eli Gafni, Sergio Rajsbaum, and Maurice Herlihy. Subconsensus tasks: Renaming is weaker than set agreement. In *Proceedings of the 20th international conference on Distributed computing (DISC’06)*, volume 4167 of *LNCS*, pages 329–338, Berlin, Heidelberg, 2006. Springer-Verlag.
- [27] Maurice Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, January 1991.
- [28] Maurice Herlihy, Dmitry Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.
- [29] Maurice Herlihy and Sergio Rajsbaum. The topology of shared-memory adversaries. In *PODC ’10: Proceeding of the 29th ACM symposium on Principles of distributed computing*, pages 105–113, New York, NY, USA, 2010. ACM.
- [30] Maurice Herlihy and Sergio Rajsbaum. The topology of distributed adversaries. *Distributed Computing*, 26(3):173–192, 2013.
- [31] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999.
- [32] Maurice P. Herlihy and Jeannette M. Wing. Linearizability: a correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12(3):463–492, July 1990.
- [33] M. C. Loui and H. H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Parallel and Distributed Computing, Advances in Computing Research. F. P. Preparata ed., JAI Press, Greenwich, CT*, 4:163–183, 1987.
- [34] Yoram Moses and Sergio Rajsbaum. A layered analysis of consensus. *SIAM J. Comput.*, 31(4):989–1021, 2002.
- [35] Sergio Rajsbaum. Iterated shared memory models. In Alejandro López-Ortiz, editor, *LATIN 2010: Theoretical Informatics*, volume 6034 of *Lecture Notes in Computer Science*, pages 407–416. Springer Berlin / Heidelberg, 2010.
- [36] Sergio Rajsbaum, Michel Raynal, and Corentin Travers. An impossibility about failure detectors in the iterated immediate snapshot model. *Inf. Process. Lett.*, 108(3):160–164, 2008.
- [37] Sergio Rajsbaum, Michel Raynal, and Corentin Travers. The iterated restricted immediate snapshot model. In *COCOON ’08: Proceedings of the 14th annual international conference on Computing and Combinatorics*, pages 487–497, Berlin, Heidelberg, 2008. Springer-Verlag.

- [38] Michael Saks and Fotios Zaharoglou. Wait-free  $k$ -set agreement is impossible: The topology of public knowledge. *SIAM J. Comput.*, 29(5):1449–1483, 2000.

## A Appendix: The proofs of Section 4.1

Here we prove Theorem 4.1 and prove the correctness of the protocol in Figure 8. First, we introduce the formal definition of the  $g$ -2coalitions-consensus task for  $g$  processes. To introduce formally the 2coalitions-consensus task, we need to define the set of all valid input values for this new task.

Let  $I$  be a non-empty set of names and  $\mathcal{N} = I \cup \{\perp\}$  ( $\perp \notin I$ ). If  $x = \langle v_1, v_2 \rangle \in \mathcal{N} \times \mathcal{N}$ ,  $x.left$  denotes the value  $v_1$  and  $x.right$  is the value  $v_2$ . An ordered  $g$ -tuple  $C = (x_1, \dots, x_g) \in (\mathcal{N} \times \mathcal{N})^g$  is a  $g$ -coalitions tuple if and only if

- (a) For all  $x_i$  in  $C$ ,  $x_i.left \neq \perp$  or  $x_i.right \neq \perp$ .
- (b) If  $x_i, x_j$  are members of  $C$  such that  $x_i.left \neq \perp$  and  $x_j.left \neq \perp$ , then  $x_i.left = x_j.left$ . A similar rule must hold if  $x_i.right \neq \perp$  and  $x_j.right \neq \perp$ .

Let  $l(C) = \{i \in \bar{g} \mid x_i.left \neq \perp\}$  and  $r(C) = \{j \in \bar{g} \mid x_j.right \neq \perp\}$ . Then we also require that

- (c)  $|l(C) - r(C)| = 1$  and  $r(C) - l(C) = \{g\}$ .

Notice that property (a) implies that  $l(C) \cup r(C) = \bar{g}$ . The set of all  $g$ -coalitions tuples is denoted by  $\mathcal{C}_g$ . We can now define a new distributed task.

**The  $g$ -2coalitions-consensus task** We have  $g$  processes  $p_1, \dots, p_g$  and each process  $p_i$  starts with a private input value of the form  $x_i \in \mathcal{N} \times \mathcal{N}$  such that  $C = (x_1, \dots, x_g) \in \mathcal{C}_g$ , and  $C$  is called a *global input*. In any execution, Termination and Agreement must be satisfied, and in addition:

- 2coalitions-Validity: If some process outputs  $c$ , then there must exists a process  $p_j$  with input  $x_j$  such that  $x_j = \langle c, v \rangle$  or  $x_j = \langle v, c \rangle$  ( $c \in I, v \in \mathcal{N}$ ).

### A.1 The consensus protocol

The formal spec of the iterated consensus protocol with safe-consensus objects is given in Figure 8. This is a protocol that implements consensus using only  $\binom{n}{2}$  2coalitions-consensus tasks. If we suppose that the protocol is correct, then we can use the  $g$ -2coalitions-consensus protocol presented in Section 4.1 (Figure 5) to replace the call to the 2coalitions-consensus objects in Figure 8 to obtain a full iterated protocol that solves the consensus task using  $\binom{n}{2}$  safe-consensus objects.

We now give a short description of how the protocol works. There are precisely  $\binom{n}{2}$  rounds executed by the processes and in each round, some subset of processes try to agree in a new consensus value among two given input values. The local variables *step*, *firstid* and *lastid* are used by the processes to store information that tell them which is the current set of processes that must try to reach a new agreement in the current round, using a 2coalitions-consensus object (the symbol "*id*" contains the id of the process which is executing the code). The local array *agreements* contains enough local registers used by the processes to store the agreements made in each round of the protocol and two distinct processes can have different agreements in *agreements*. Each consensus value  $v$  stored in a register of *agreements* is associated with two integers  $i_1, i_r \in \bar{n}$  ( $r \geq 1$ ), which represent a set of processes  $p_{i_1}, \dots, p_{i_r}$  (with  $i_1 < \dots < i_r$ ) that have invoked a

```

(1) init  $step, firstid, lastid \leftarrow 1$ ;
       $C, D, dec, newagreement \leftarrow \perp$ ;
       $agreements[\mathbf{tup}(id, id)] \leftarrow input$ ;

(2) begin
(3)   for  $r \leftarrow 1$  to  $\binom{n}{2}$ 
(4)      $lastid \leftarrow firstid + step$ ;
(5)     if  $firstid \leq id \leq lastid$  then
(6)        $C \leftarrow agreements[\mathbf{tup}(firstid, lastid - 1)]$ ;
(7)        $D \leftarrow agreements[\mathbf{tup}(firstid + 1, lastid)]$ ;
(8)        $newagreement \leftarrow 2coalitions-consensus[r](C, D)$ ;
(9)        $agreements[\mathbf{tup}(firstid, lastid)] = newagreement$ ;
(10)    end if
(11)    if  $lastid < n$  then
(12)       $firstid = firstid + 1$ ;
(13)    else if  $firstid > 1$  then
(14)       $firstid = 1$ ;
(15)       $step = step + 1$ ;
(16)    end if
(17)  end for
(18)   $dec \leftarrow agreements[\mathbf{tup}(1, n)]$ ;
(19)  decide  $dec$ ;
(20) end

```

Figure 8: An iterated consensus protocol using  $g$ -2coalitions-consensus objects.

2coalitions-consensus object to agree on the value  $v$ , thus we can say that  $v$  is an agreement made by the coalition of processes represented by the pair  $(i_1, i_r)$ . To be able to store  $v$  in the array *agreements*, the processes use a deterministic function  $\mathbf{tup}: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ , which maps bijectively  $\mathbb{N} \times \mathbb{N}$  onto  $\mathbb{N}$ . This map can be easily constructed, for example, here is a simple definition for  $\mathbf{tup}$

$$\mathbf{tup}(i, j) = \binom{i + j + 1}{2} + j. \quad (5)$$

Using all the elements described above, the processes can use the protocol of Figure 8 to implement consensus using  $\binom{n}{2}$  2coalitions-consensus objects, building in the process the structure depicted in Figure 4 from Section 4.1. From the first round and up to round  $n - 1$ , all the processes use their proposed input values to make new agreements in pairs, in round one,  $p_1, p_2$  invoke a 2coalitions-consensus shared object to agree on a common value, based on their input values; in round 2,  $p_2$  and  $p_3$  do the same with another 2coalitions-consensus object and their own input values; later, the turn comes to  $p_3$  and  $p_4$  to do the same with another shared object and their input values and so on until the round number  $n - 1$ , where  $p_{n-1}$  and  $p_n$  agree on a common value in the way that we have already described. All the agreements obtained in these  $n - 1$  rounds are stored on the local registers

$$\begin{aligned}
& agreements_1[\mathbf{tup}(1, 2)], agreements_2[\mathbf{tup}(1, 2)], agreements_2[\mathbf{tup}(2, 3)], \\
& agreements_3[\mathbf{tup}(2, 3)], agreements_3[\mathbf{tup}(3, 4)], agreements_4[\mathbf{tup}(3, 4)], \\
& \quad \dots, \\
& agreements_{n-1}[\mathbf{tup}(n-1, n)], agreements_n[\mathbf{tup}(n-1, n)]. \quad (6)
\end{aligned}$$

In this way, the processes build the first part of the structure shown in Figure 4 of Section 4.1. At the end of round  $n-1$ , in lines 14-15, each process updates the values of the local variables *step* and *firstid* and proceeds to round  $n$ . What happens in the next  $n-2$  rounds, is very similar to the case of the previous  $n-1$  rounds, but instead of making agreements in pairs, the processes reach new agreements in groups of three processes (see Figure 4) invoking new 2coalitions-consensus shared objects and the consensus values obtained in the first  $n-1$  rounds and when each process reaches round  $n-1+n-2=2n-3$ , it updates the values of its local variables *step* and *firstid* and then the processes proceed to round  $2n-2$ .

In general, when the processes are about to begin executing round  $(\sum_{j=1}^m n-j)+1$  ( $m < n$ ), they will try to make  $n-(m+1)$  new agreements in groups of size  $m+2$ , with the aid of the 2coalitions-consensus objects and the agreements they obtained from the previous  $n-m$  rounds and store the new consensus values in their local arrays *agreements*, using the **tup** function. When a process finishes round  $(\sum_{j=1}^{m+1} n-j)$ , the values of *step* and *firstid* are updated to repeat the described behavior for the next  $n-(m+2)$  rounds, until the  $\binom{n}{2}$  round, where the last agreement is made and this value is the output value of all the processes.

Now we are ready to give the full proof of Theorem 4.1. We need a series of Lemmas.

**Lemma A.1.** *The protocol in Figure 8 solves the consensus task using  $\binom{n}{2}$   $g$ -2coalitions-consensus objects.*

*Proof.* The protocol clearly satisfies the Termination condition of the consensus task (the only loop is finite). We prove that it fulfills the Agreement and Validity conditions; to do this, we need some definitions and some intermediate results.

With respect to the protocol of Figure 8, let  $F = \{i_1, \dots, i_r\} \subseteq \bar{n}$  be elements such that  $i_1 < \dots < i_r$ . The set  $F$  will be called a *coalition* and will be denoted by  $F = [i_1, \dots, i_r]$ . Let  $p_i$  a process such that  $i \in F$ . We say that  $p_i$  belongs to the coalition  $F$  if and only if the following condition is fulfilled:

- (1)  $agreements_i[\mathbf{tup}(i_1, i_r)] = v$ , where  $v \neq \perp$  is a valid input value proposed by some participating process.

We ask the following *agreement* property to be satisfied by  $F$ :

- (2) if  $p_i, p_j$  are processes which belong to  $F$  and  $v, v'$  are the values which satisfy (1) for  $p_i$  and  $p_j$  respectively, then  $v = v'$ .

The value  $v$  of the first condition is called the *name* of the coalition. The second condition says that all processes that belong to the same coalition must agree on the coalition's name. For  $n > m \geq 0$ , let  $\gamma(n, m)$  be defined by

$$\gamma(n, m) = \begin{cases} 0 & \text{if } m = 0, \\ \gamma(n, m-1) + n - m & \text{if } m > 0. \end{cases}$$

Notice that  $\gamma(n, m) = \sum_{i=1}^m n - i$  for  $m > 0$  and  $\gamma(n, n-1) = \binom{n}{2}$ .

**Lemma A.2.** *Let  $1 \leq m \leq n-1$  and  $1 \leq c \leq n-m$ . If  $p_i$  is executing the protocol at the beginning of round number  $r = \gamma(n, m-1) + c$  (before line 11) then  $step_i = m$  and  $firstid_i = c$ .*

*Proof.* We prove this by induction on  $m$ . An easy analysis of the code in Figure 8 shows that the base case holds (when  $m = 1$ , in the first  $n-1$  rounds). Assume that for  $m < n-1$ , the lemma is true. We first prove the following claim: When  $p_i$  starts executing round  $\gamma(n, m) + 1$ ,  $step_i = m+1$  and  $firstid_i = 1$ . By the induction hypothesis, when process  $p_i$  executed the protocol at round  $r' = \gamma(n, m) = \gamma(n, m-1) + c = \gamma(n, m-1) + (n-m)$  before line 11, the local variables  $step_i$  and  $firstid_i$  had the values of  $m$  and  $n-m$  respectively. When  $p_i$  reached line 11, it executed the test of the **if** statement, but before that,  $p_i$  executed line 4 of the protocol, gather that,  $lastid_i = firstid_i + step_i = (n-m) + m = n$ ; thus the test in line 11 failed and  $p_i$  executed lines 14-15 of the **else** statement ( $firstid_i > 1$  because  $m < n-1$ ) and then  $step_i$  was incremented by one and  $firstid_i$  was set to 1 at the end of round  $r'$ . Therefore, when  $p_i$  starts executing round  $\gamma(n, m) + 1$ ,  $step_i = m+1$  and  $firstid_i = 1$ .

Now suppose that  $p_i$  executes the protocol at the beginning of round number  $r = \gamma(n, m) + c$  where  $1 \leq c \leq n-(m+1)$ . If  $c = 1$  then by the preceding argument,  $step_i = m+1$  and  $firstid_i = 1 = c$ . Using this as a basis for an inductive argument, we can prove that for  $c \in \{1, \dots, n-(m+1)\}$ ,  $c = firstid_i$  and  $step_i = m+1$ .  $\square$

**Lemma A.3.** *Let  $0 \leq m \leq n-2$ . Suppose that process  $p_i$  is about to begin executing the protocol at round  $r = \gamma(n, m) + c$  ( $1 \leq c \leq n-(m+1)$ ). If  $c \leq i \leq c+m+1$  and  $p_i$  belongs to the coalition  $P = [c, \dots, c+m]$  or it belongs to the coalition  $Q = [c+1, \dots, c+m+1]$ , then at the end of round  $r$ ,  $p_i$  belongs to the coalition  $[c, \dots, c+m+1]$ .*

*Proof.* Let  $p_i$  be any process that begins executing round  $r$ . By Lemma A.2, we know that  $step_i = m+1$  and  $firstid_i = c$ , which implies that  $lastid_i = c+m+1$ . If  $i \in \{c, \dots, c+m+1\}$ , the **if**'s test at line 5 is successful and after that what happens in lines 6,7 depends on  $i$ . If  $c \leq i \leq c+m$ , then  $p_i$  is in the coalition  $P$  and if  $c+1 \leq i \leq c+m+1$ ,  $p_i$  is in the coalition  $Q$ , gather that, a valid input value is assigned to at least one of the variables  $C_i$  or  $D_i$ , so  $p_i$  invokes in line 8 the  $(m+2)$ -2coalitions-consensus object with the input  $x_i = \langle C_i, D_i \rangle$ . We now pause for a moment to argue that the tuple  $J = (x_c, \dots, x_{c+m+1})$  build with the inputs of processes  $p_c, \dots, p_{c+m+1}$  is a valid global input for the  $(m+2)$ -2coalitions-consensus task. Indeed, from the hypothesis, it is easy to see that  $J$  satisfies the requirements (a)-(c) of the definition of coalition tuple and notice that  $r(J) - l(J) = \{c+m+1\}$  and  $l(J) - r(J) = \{c\}$ .

Now back to the execution of process  $p_i$ . After  $p_i$  invokes the  $(m+2)$ -2coalitions-consensus task, the return value of the shared object, say  $v$ , is assigned to the local variable  $newagreement_i$ . Finally,  $p_i$  stores the contents of  $newagreement_i$  in the local array  $agreements_i$  at position **tup**( $firstid_i, lastid_i$ ), where  $firstid_i = c$  and  $lastid_i = c+m+1$ . Because of the Agreement condition of the 2coalitions-consensus task, every process with id in the set  $\{c, \dots, c+m+1\}$  obtained the same value  $v$  as return value from the shared object and by the 2coalitions-Validity, this is a valid input value proposed by some process. Therefore process  $p_i$  belongs to the coalition  $[c, \dots, c+m+1]$  at the end of round  $r$ .

On the other hand, if  $i \notin \{c, \dots, c+m+1\}$ ,  $p_i$  does not executes the body of the **if** statement (lines 6 to 9) and it goes on to execute the **if/else** block at lines 11-16 and then round  $r$  ends for  $p_i$ , thus it does not try to make a new coalition with other processes.  $\square$



**Lemma A.4.** *Let  $m \in \{0, \dots, n-2\}$  be fixed. Suppose that process  $p_j$  has executed the protocol for  $\gamma(n, m)$  rounds and that there exists  $i \in \{1, \dots, n - (m+1)\}$  such that  $p_j$  belongs to the coalition  $[i, \dots, i+m]$  or it belongs to the coalition  $[i+1, \dots, i+m+1]$ . Then after  $p_j$  has executed the protocol for  $n - (m+1)$  more rounds,  $p_j$  belongs to the coalition  $[i, \dots, i+m+1]$ .*

*Proof.* We apply Lemma A.3 in each round  $\gamma(n, m) + c$ , where  $c \in \{1, \dots, n - (m+1)\}$ .  $\square$

Now we can complete the proof of Lemma A.1. Let  $p_j$  be any process that executes the protocol. Just at the beginning of the first round (line 2), process  $p_j$  belongs to the coalition  $[j]$  (because of the assignments made to the local array *agreements<sub>j</sub>* in line 1, so that if  $p_j$  executes the protocol for  $\gamma(n, 1) = n - 1$  rounds, we can conclude using Lemma A.4, that process  $p_j$  belongs to some of the coalitions  $[i, i+1]$  ( $1 \leq i \leq n-1$ ). Starting from this fact and using induction on  $m$ , we can prove that for all  $m = 1, \dots, n-1$ ; at the end of round  $\gamma(n, m)$ ,  $p_j$  belongs to some of the coalitions  $[i, \dots, i+m]$  ( $1 \leq i \leq n-m$ ). In the last round (when  $m = n-1$ ), after executing the main **for** block, process  $p_j$  belongs to the coalition  $T = [1, \dots, n]$ , thus when  $p_j$  executes line 18, it will assign to the local variable *dec<sub>i</sub>* a valid proposed input value and this is the value decided by  $p_j$  at line 19. All processes decide the same value because all of them are in the coalition  $T$ . Therefore the protocol satisfies the Agreement and Validity conditions of the consensus task.  $\square$

## A.2 Solving 2coalition-consensus with one safe-consensus

We now prove that the  $g$ -2coalitions-consensus protocol presented in Section 4.1 (Figure 5) is correct.

**Lemma A.5.** *The protocol of Figure 5 solves the  $g$ -2coalitions-consensus task using one snapshot object and one safe-consensus object.*

*Proof.* Let  $p_i \in \{p_1, \dots, p_g\}$ ; after  $p_i$  writes the tuple  $\langle v_1, v_2 \rangle$  to the snapshot object  $SM$ , it invokes the safe-consensus object and takes a snapshot of the shared memory,  $p_i$  enters into the **if/else** block at lines 6-10. Suppose that the test in line 6 is successful, this says that the safe-consensus object returned the value  $g$  to process  $p_i$ . By the Safe-Validity condition of the safe-consensus task, either process  $p_g$  invoked the shared object or at least two processes  $p_j, p_k$  invoked the safe-consensus object concurrently (and it could happen that  $j, k \neq g$ ). In any case, the participating processes wrote their input tuples to the snapshot object  $SM$  before accessing the safe-consensus object, so that  $p_i$  can see these values in its local variable *sm<sub>i</sub>*, and remember that the coalitions tuple  $C$  consisting of all the input values of processes  $p_1, \dots, p_g$  satisfies the properties

$$|l(C) - r(C)| = 1 \text{ and } r(C) - l(C) = \{g\}.$$

Thus, the equation  $|l(C) - r(C)| = 1$  tell us that only one process has input tuple  $\langle x, \perp \rangle$  ( $x \neq \perp$ ), then when  $p_i$  executes line 7, it will find a local register in *sm<sub>i</sub>* such that *sm<sub>i</sub>*[*j*].*right*  $\neq \perp$  and this value is assigned to *dec<sub>i</sub>*. Finally,  $p_i$  executes line 11, where it decides the value stored in *dec<sub>i</sub>* and this variable contains a valid input value proposed by some process.

If the test at line 6 fails, the argument to show that *dec<sub>i</sub>* contains a valid proposed input value is very similar to the previous one. This proves that the protocol satisfies the 2coalitions-Validity condition of the  $g$ -2coalitions-consensus task.

The Agreement condition is satisfied because by the Agreement condition of the safe-consensus task, all participating processes receive the same output value from the shared object and therefore

all processes have the same value in the local variables  $val_i$  ( $1 \leq i \leq g$ ), which means that all participating processes execute line 7 or all of them execute line 9. Then, for every process  $p_r$ ,  $dec_r$  contains the value  $sm_r[j_r].right$  or the value  $sm_r[j'_r].left$  where  $j_r$  ( $j'_r$ ) depend on  $p_r$ . But because the input pairs of the processes constitute a coalitions tuple  $C$ , they satisfy property (b) of the definition of coalitions tuple, which implies that  $sm_r[j_r].right = sm_r[j_q].right$  whenever  $sm_r[j_r].right$  and  $sm_r[j_q].right$  are non- $\perp$  (a similar statement holds for the left sides). We conclude that all processes assign to the variables  $dec_i$  ( $1 \leq i \leq g$ ) the same value and thus all of them decide the same output value, that is, the Agreement property of the  $g$ -2coalitions-consensus tasks is fulfilled. The Termination property is clearly satisfied.  $\square$

## B Appendix: Results on subgraphs of Johnson graphs

In this appendix, we prove Theorem B.7, the main combinatorial result that we need to build the proof of the lower bound given by Theorem 4.2. We will be using some combinatorial facts of graph theory and finite sets.

### B.1 Subgraphs of Johnson graphs

Our graph terminology is standard, see for example [8], all the graphs that we use are simple graphs. For any set  $A$  and  $E \subseteq 2^A$ , we denote the union of all the elements of every set in  $E$  as  $\bigcup E$ . For  $1 \leq m \leq n$ , the *Johnson graph*  $J_{n,m}$  has as vertex set all subsets of  $\bar{n}$  of cardinality  $m$ , two vertices  $b_1, b_2$  are adjacent if and only if  $|b_1 \cap b_2| = m - 1$ . Let  $V_{n,m} = V(J_{n,m})$  and  $U \subseteq V_{n,m}$ , define the set  $\zeta(U)$  as

$$\zeta(U) = \{c \cup d \mid c, d \in U \text{ and } |c \cap d| = m - 1\}. \quad (7)$$

Notice that each  $f \in \zeta(U)$  has size  $m + 1$  because, if  $f = c \cup d$  for  $c, d \in U$ , then  $|c| = |d| = m$  and it is known that  $|c \cap d| = m - 1 \Leftrightarrow |c \cup d| = m + 1$ . Thus  $\zeta(U) \subseteq V_{n,m+1}$ . For any  $v = 0, \dots, n - m$ , the *iterated  $\zeta$ -operator*  $\zeta^v$  is given by

$$\zeta^v(U) = \begin{cases} U & \text{if } v = 0, \\ \zeta(\zeta^{v-1}(U)) & \text{otherwise.} \end{cases} \quad (8)$$

As  $U \subseteq V_{n,m}$ , we can check that  $\zeta^v(U) \subseteq V_{n,m+v}$ . A simple, but useful property of the  $\zeta$ -operator is that

$$\bigcup \zeta^v(U) \subseteq \bigcup U. \quad (9)$$

### B.2 Some results on the connectivity of subgraphs of $J_{n,m}$

We are ready to prove all the combinatorial results that we need.

**Lemma B.1.** *Let  $U \subseteq V_{n,m}$  and  $G = G[U]$ . The following properties are satisfied.*

- (i) *If  $G$  is connected, then  $|\bigcup U| \leq m - 1 + |U|$ .*
- (ii) *If  $U_1, \dots, U_r$  are connected components of  $G$ , then  $\zeta(\bigcup_{i=1}^r U_i) = \bigcup_{i=1}^r \zeta(U_i)$ .*

*Proof.* (i) can be proven easily using induction on  $|U|$  and (ii) is an easy consequence of the definitions, thus we omit the proof.  $\square$

**Lemma B.2.** *Let  $U \subseteq V_{n,m}$  with  $|U| \leq n - m$ . If  $G[U]$  is connected then  $\bigcup U \neq \bar{n}$ .*

*Proof.* Suppose that with the given hypothesis  $\bigcup U = \bar{n}$ . As  $G[U]$  is connected, we can use part (i) of Lemma B.1 to obtain the inequality  $n \leq m + |U| - 1$ . But this implies that  $|U| \geq n - m + 1$  and this is a contradiction. Therefore  $\bigcup U \neq \bar{n}$ .  $\square$

The following lemma is about subgraphs of  $J_{n,2}$ , it is a small result needed to give the full proof of our lower bound (see Lemma C.10). However, this result is far easier to prove than Theorem B.7.

**Lemma B.3.** *Let  $U \subset V_{n,2}$  with  $|U| \leq n - 2$ . Then there exists a partition  $\bar{n} = A \cup B$  such that*

$$(\forall b \in U)(b \subseteq A \text{ or } b \subseteq B).$$

*Proof.* If  $\bigcup U \neq \bar{n}$ , then setting  $A = \bar{n} - \{i\}$  and  $B = \{i\}$  where  $i \notin \bigcup U$  we are done. Otherwise, by Lemma B.2, the induced subgraph  $G = G[U]$  of  $J_{n,2}$  is disconnected. There exists a partition  $V_1, V_2$  of  $V(G) = U$  with the property that there is no edge of  $G$  from any vertex of  $V_1$  to any vertex of  $V_2$ . Let

$$A = \bigcup V_1 \quad \text{and} \quad B = \bigcup V_2.$$

It is easy to show that  $\bar{n} = A \cup B$  is the partition of  $\bar{n}$  that we need.  $\square$

**Lemma B.4.** *Let  $U \subset V_{n,m}$  with  $|U| > 1$  and  $G = G[U]$  a connected subgraph of  $J_{n,m}$ . Then the graph  $G' = G[\zeta(U)]$  is connected and  $\bigcup \zeta(U) = \bigcup U$ .*

*Proof.* We show that  $G'$  is connected. If  $G$  contains only two vertices, then  $G'$  contains only one vertex and the result is immediate. So assume that  $|U| > 2$  and take  $b, c \in \zeta(U)$ . We need to show that there is a path from  $b$  to  $c$ . We know that  $b = b_1 \cup b_2$  and  $c = c_1 \cup c_2$  with  $b_i, c_i \in U$  for  $i = 1, 2$ . As  $G$  is connected, there is a path

$$v_1 = b_2, v_2, \dots, v_q = c_2,$$

and we use it to build the following path in  $G'$ ,

$$b = b_1 \cup v_1, v_1 \cup v_2, \dots, v_{q-1} \cup v_q, c = v_q \cup c_1,$$

thus  $b$  and  $c$  are connected in  $G'$ , so that it is a connected graph.

Now we prove that  $\bigcup \zeta(U) = \bigcup U$ . By Equation (9),  $\bigcup \zeta(U) \subseteq \bigcup U$ , so it only remains to prove the other inclusion. Let  $x \in \bigcup U$ , there is a vertex  $b \in U$  such that  $x \in b$  and as  $|U| > 1$  and  $G$  is connected, there exists another vertex  $c \in U$  such that  $b$  and  $c$  are adjacent in  $G$ . Then  $b \cup c \in \zeta(U)$ , thus

$$x \in b \subseteq b \cup c \subseteq \bigcup \zeta(U),$$

gather that,  $\bigcup U \subseteq \bigcup \zeta(U)$  and the equality holds. This concludes the proof.  $\square$

**Lemma B.5.** *Let  $U \subseteq V_{n,m}$ ,  $G = G[U]$ ,  $G' = G[\zeta(U)]$  and  $\mathcal{U}$  be the set of connected components of the graph  $G$ . Then the following conditions hold:*

1. *For any connected component  $V$  of  $G'$ , there exists a set  $\mathcal{O} \subseteq \mathcal{U}$  such that  $V = \zeta(\bigcup \mathcal{O}) = \bigcup_{Z \in \mathcal{O}} \zeta(Z)$ .*

2. If  $V, V'$  are two different connected components of  $G'$  and  $\mathcal{O}, \mathcal{O}' \subseteq \mathcal{U}$  are the sets which fulfill property 1 for  $V$  and  $V'$  respectively, then  $\mathcal{O} \cap \mathcal{O}' = \emptyset$ .

*Proof.* Part 2 is clearly true, so we only need to prove part 1. Define the graph  $\mathcal{H} = (V(\mathcal{H}), E(\mathcal{H}))$  as follows:

$$V(\mathcal{H}) = \mathcal{U};$$

Two vertices  $Z, Z'$  form an edge in  $E(\mathcal{H})$  if and only if there exist  $b, c \in Z$  and  $d, e \in Z'$  such that

- $|b \cap c| = m - 1$  and  $|d \cap e| = m - 1$
- $|(b \cup c) \cap (d \cup e)| \geq m$ .

Roughly speaking,  $\mathcal{H}$  describes for two connected components  $Z, Z'$  of  $G$ , whether  $\zeta(Z), \zeta(Z')$  lie in the same connected component of  $G'$  or not. Let  $V$  be a connected component of  $G'$ , if  $b \in V$ , then  $b = b_1 \cup b_2$ , where  $b_1, b_2$  are in some connected component  $Z_b \subseteq U$  of  $G$ . In  $\mathcal{H}$ , there exists a component  $\mathcal{O}$  such that  $Z_b \in \mathcal{O}$ . By Lemma B.4,  $G[\zeta(Z_b)]$  is connected and  $b \in \zeta(Z_b) \cap V$ , so it is clear that  $\zeta(Z_b) \subseteq V$ . Suppose that  $Z' \in \mathcal{O}$  with  $Z' \neq Z_b$  and these components form an edge in  $\mathcal{H}$ . This means that  $G[\zeta(Z_b)]$  and  $G[\zeta(Z')]$  are joined by at least one edge in  $G'$ , thus every vertex of  $\zeta(Z')$  is connected with every vertex of  $\zeta(Z_b)$ , and as  $\zeta(Z_b) \subseteq V$ ,  $\zeta(Z_b) \cup \zeta(Z') \subseteq V$ . We can continue this process with every element of the set  $\mathcal{O} - \{Z_b, Z'\}$  to show that

$$\zeta(\bigcup \mathcal{O}) = \bigcup_{Z \in \mathcal{O}} \zeta(Z) \subseteq V.$$

(The first equality comes from part (ii) of Lemma B.1). We prove the other inclusion, if  $V = \{b\}$ , we are done. Otherwise, let  $c \in V - \{b\}$ , if  $c \in \zeta(Z_b)$ , then  $c \in \zeta(\bigcup \mathcal{O})$ . In case that  $c \notin \zeta(Z_b)$ , there must exist some connected component  $Z_c$  of  $G$  such that  $c \in \zeta(Z_c)$  and  $Z_c \neq Z_b$ . In  $G'$ , we can find a path  $b = v_1, \dots, v_q = c$  where  $v_i \in V$  for  $i = 1, \dots, q$ . The set  $\mathcal{Q} \subset \mathcal{U}$  defined as

$$\mathcal{Q} = \{X \in \mathcal{U} \mid (\exists j)(1 \leq j \leq q \text{ and } v_j \in \zeta(X))\},$$

can be seen to have the property that every pair of vertices  $X, X' \in \mathcal{Q}$  are connected by a path in  $\mathcal{H}$ . Since  $Z_b, Z_c \in \mathcal{Q}$ , they are connected in  $\mathcal{H}$  and as  $Z_b \in \mathcal{O}$ , then  $Z_c \in \mathcal{O}$ , gather that,  $c \in \zeta(\bigcup \mathcal{O})$ . Therefore  $V \subseteq \zeta(\bigcup \mathcal{O})$  and the equality  $V = \bigcup_{Z \in \mathcal{O}} \zeta(Z)$  holds. This proves part 1 and finishes the proof.  $\square$

**Lemma B.6.** Let  $U \subseteq V_{n,m}$  and  $\mathcal{U}$  be the set of connected components of  $G[U]$ . Then for all  $s \in \{0, \dots, n - m\}$  and  $G^s = G[\zeta^s(U)]$ , the following conditions are satisfied.

**H1** For every connected component  $V$  of the graph  $G^s$ , there exists a set  $\mathcal{O} \subseteq \mathcal{U}$  such that

$$|\bigcup V| \leq m - 1 + \sum_{Z \in \mathcal{O}} |Z|. \quad (10)$$

**H2** If  $V, V'$  are two different connected components of  $G^s$  and  $\mathcal{O}, \mathcal{O}' \subseteq \mathcal{U}$  are the sets which make true the inequality given in (10) for  $V$  and  $V'$  respectively, then  $\mathcal{O} \cap \mathcal{O}' = \emptyset$ .

*Proof.* We prove the lemma by using induction on  $s$ . For the base case  $s = 0$ ,  $G^0 = G[U]$ , we use Lemma B.1 and we are done. Suppose that for  $0 \leq s < n - m$ , **H1** and **H2** are true. We prove the case  $s + 1$ , let  $W$  be a connected component of the graph  $G^{s+1}$ . By part 1 of Lemma B.5, we know that there is a unique set  $\mathcal{Q}$  of connected components of  $G^s$  such that

$$W = \bigcup_{Q \in \mathcal{Q}} \zeta(Q)$$

and because  $\zeta(Q) \neq \emptyset$ ,  $|Q| > 1$ , so that by Lemma B.4,  $\bigcup \zeta(Q) = \bigcup Q$  for all  $Q \in \mathcal{Q}$ , thus it is true that

$$\begin{aligned} \bigcup W &= \bigcup_{Q \in \mathcal{Q}} (\bigcup \zeta(Q)) \\ &= \bigcup_{Q \in \mathcal{Q}} (\bigcup Q) \\ &= \bigcup_{Q \in \mathcal{Q}} Q. \end{aligned}$$

By the induction hypothesis,  $|\bigcup Q| \leq m - 1 + \sum_{Z \in \mathcal{O}_Q} |Z|$ , where  $\mathcal{O}_Q \subseteq \mathcal{U}$  for all  $Q \in \mathcal{Q}$  and when  $Q \neq R$ ,  $\mathcal{O}_Q \cap \mathcal{O}_R = \emptyset$ . With a simple induction on  $|\mathcal{Q}|$ , it is rather forward to show that

$$|\bigcup W| \leq m - 1 + \sum_{Z \in \mathcal{O}} |Z|,$$

where  $\mathcal{O} = \bigcup_{Q \in \mathcal{Q}} \mathcal{O}_Q$ , gather that, property **H1** is fulfilled. We now show that condition **H2** holds. If  $W'$  is another connected component of  $G^{s+1}$ , then applying the above procedure to  $W'$  yields  $W' = \bigcup_{S \in \mathcal{S}} \zeta(S)$ ,  $\bigcup W' = \bigcup_{S \in \mathcal{S}} (\bigcup S)$  and  $|\bigcup W'| \leq m - 1 + \sum_{X \in \mathcal{O}'} |X|$ , with  $\mathcal{O}' = \bigcup_{S \in \mathcal{S}} \mathcal{O}_S$ . By 2 of Lemma B.5,  $\mathcal{Q} \cap \mathcal{S} = \emptyset$ , so that if  $Q \in \mathcal{Q}$  and  $S \in \mathcal{S}$ , then  $Q \neq S$  and the induction hypothesis tell us that  $\mathcal{O}_Q \cap \mathcal{O}_S = \emptyset$ , thus

$$\begin{aligned} \mathcal{O} \cap \mathcal{O}' &= \left( \bigcup_{Q \in \mathcal{Q}} \mathcal{O}_Q \right) \cap \left( \bigcup_{S \in \mathcal{S}} \mathcal{O}_S \right) \\ &= \bigcup_{(Q,S) \in \mathcal{Q} \times \mathcal{S}} (\mathcal{O}_Q \cap \mathcal{O}_S) \\ &= \emptyset, \end{aligned}$$

and property **H2** is satisfied, so that by induction we obtain the result.  $\square$

We can now prove the main result of the combinatorial part of the proof of Theorem 4.2.

**Theorem B.7.** *Let  $U \subset V_{n,m}$  such that  $|U| \leq n - m$ . Then  $\zeta^{n-m}(U) = \emptyset$ .*

*Proof.* For a contradiction, suppose that  $\zeta^{n-m}(U) \neq \emptyset$ , then  $G[\zeta^{n-m}(U)]$  is the graph  $J_{n,n}$  and contains the unique connected component  $C = \{\bar{n}\}$ , thus by Lemma B.6, for some set  $\mathcal{O}$  of connected components of  $G[U]$ ,

$$n = |\bigcup C| \leq m - 1 + \sum_{Z \in \mathcal{O}} |Z| \leq m - 1 + |U|,$$

and we conclude that  $|U| \geq n - m + 1$ , a contradiction. So that  $\zeta^{n-m}(U) \neq \emptyset$  is impossible. Therefore  $\zeta^{n-m}(U)$  has no elements.  $\square$

## C Appendix: The proofs of Section 4.2

In this last appendix, we give the required elements to prove Theorem 4.2. We prove Lemmas C.8 and C.10, these results give us information about the structure of WOR protocols with safe-consensus objects and the combinatorial interactions that exists between the sets of processes which cannot distinguish between global states of a path and the boxes that represent safe-consensus objects invoked by the processes. For all this, we need to apply Lemma B.3 and Theorem B.7. We will be using definitions and terminology given in Sections 2, 3, 4.2 and Appendix B.

Before giving the formal proof of our results, we need some technical definitions which will be useful to prove our results. Let  $\mathcal{A}$  be a protocol with safe-consensus objects for  $n$ -processes,  $X \subseteq \bar{n}$ ,  $R$  a state of some round  $r > 0$  and  $b \in \mathbf{Inv}(R)$ . We say that  $R$  is a *ladder state for  $X$  with step  $b$* , if  $R = S \cdot \xi(C_1, \dots, C_u, B)$  ( $u \geq 0$ ), where  $S$  is a reachable state in  $\mathcal{A}$  and

- $X = (\bigcup_j^u C_j) \cup B$ ;
- for  $j = 1, \dots, u$ ,  $0 \leq |C_j| \leq 2$ ;
- $(\bigcup_j^u C_j) \cap b = \emptyset$ ;
- $B = b \cap X$ .

The following definition is given for convenience. For each box  $b_i$  and  $W \subseteq \bar{n}$ , let the set  $W_{(i)}$  be defined as

$$W_{(i)} = W \cap b_i. \quad (11)$$

### C.1 The structure of WOR protocols with safe-consensus objects

We are ready to give the structural results about WOR protocols with safe-consensus objects.

**Lemma C.1.** *Let  $\mathcal{A}$  be a WOR protocol for  $n \geq 2$  processes with safe-consensus objects and  $S$  a reachable state in  $\mathcal{A}$ . Then for any two round schedules  $\pi_1, \pi_2$ ,  $\mathbf{Inv}(S \cdot \pi_1) = \mathbf{Inv}(S \cdot \pi_2)$ .*

*Proof.* In Figure 1, notice that process  $p_i$  executes the deterministic function  $h$  with the local state it has from the previous round (except from the round counter  $r$ ), so that if  $r$  is the round number of  $S$  and  $p_i$  starts executing the protocol  $\mathcal{A}$  in round  $r + 1$  with the local state it had in  $S$ , the input to the function  $h$  is given by the tuple  $(r + 1, i, sm_S, val_S)$ , where  $sm_S, val_S$  depend only on  $S$ . Thus in both successor states  $S \cdot \pi_1$  and  $S \cdot \pi_2$ ,  $p_i$  feeds the same input to  $h$ , gather that

$$\mathbf{Inv}(S \cdot \pi_1) = \mathbf{Inv}(S \cdot \pi_2)$$

and the lemma is true. □

**Lemma C.2.** *Let  $n \geq 2$ ,  $\emptyset \neq X \subseteq \bar{n}$ ,  $\mathcal{A}$  a WOR protocol with safe-consensus objects,  $S$  a state that is reachable in  $\mathcal{A}$  in some round  $r \geq 0$  and  $b_j \in \mathbf{Inv}(S_X)$ , where  $S_X = S \cdot \xi(X)$ . Then there exists a state  $L$ , such that  $L$  is a ladder state for  $X$  with step  $b_j$ , such that  $S_X$  and  $L$  are connected in round  $r + 1$  with a path of states  $\mathbf{p}$  with  $\deg \mathbf{p} \geq n - 2$ .*

*Proof.* Let  $\mathbf{Inv}(S_X) = \{b_1, \dots, b_q\}$  ( $q \geq 1$ ). By Lemma C.1, for any one-round successor state  $Q$  of  $S$ ,  $\mathbf{Inv}(Q) = \mathbf{Inv}(S_X)$  and we can write  $\mathbf{Inv}_S$  instead of  $\mathbf{Inv}(S_X)$ . Without loss, assume that  $b_1 = b_j$ . If  $q = 1$ , then  $b_1 = \bar{n}$  and the result is immediate, because  $S_X$  is a ladder state for  $X$

with step  $\bar{n}$ . So that we suppose that  $q > 1$ . Partition  $X$  as  $X = X_{(1)} \cup \dots \cup X_{(q)}$  and build the following sequence of connected states

$$\begin{aligned}
& S_X \xrightarrow{\bar{n}-\Lambda_2(1)} S \cdot \xi(\Lambda_2(1), X - \Lambda_2(1)) \\
& \quad \xrightarrow{\bar{n}-\Lambda_2(2)} S \cdot \xi(\Lambda_2(1), \Lambda_2(2), X - (\Lambda_2(1) \cup \Lambda_2(2))) \\
& \quad \quad \xrightarrow{\bar{n}-\Lambda_2(3)} \dots \xrightarrow{\bar{n}-\Lambda_2(\alpha_2)} \\
& \quad \quad S \cdot \xi(\Lambda_2(1), \Lambda_2(2), \dots, \Lambda_2(\alpha_2), X - X_{(2)}), \quad (12)
\end{aligned}$$

where  $X_{(2)} = \bigcup_{i=1}^{\alpha_2} \Lambda_2(i)$  is a partition of  $X_{(2)}$  such that  $|\Lambda_2(j)| = 1$  for  $j = 2, \dots, \alpha_2$ . The set  $\Lambda_2(1)$  has cardinality given by

$$|\Lambda_2(1)| = \begin{cases} 2 & \text{if } |X_{(2)}| > 1, \\ |X_{(2)}| & \text{otherwise,} \end{cases}$$

and we choose the safe-consensus value of every box in the set  $\mathbf{Inv}_S$  to be the same in each state of the previous sequence. This can be done because of the way we partition  $X_{(2)}$ , the election of the elements of the set  $\Lambda_2(1)$  and the properties of the safe-consensus task (Safe-Validity). We execute similar steps with box  $b_3$ , so that we obtain the path

$$\begin{aligned}
& S \cdot \xi(\Lambda_2(1), \dots, \Lambda_2(\alpha_2), X - X_{(2)}) \\
& \quad \xrightarrow{\bar{n}-\Lambda_3(1)} S \cdot \xi(\Lambda_2(1), \dots, \Lambda_2(\alpha_2), \Lambda_3(1), X - (X_{(2)} \cup \Lambda_3(1))) \\
& \quad \xrightarrow{\bar{n}-\Lambda_3(2)} S \cdot \xi(\Lambda_2(1), \dots, \Lambda_2(\alpha_2), \Lambda_3(1), \Lambda_3(2), X - (X_{(2)} \cup \Lambda_3(1) \cup \Lambda_3(2))) \\
& \quad \quad \xrightarrow{\bar{n}-\Lambda_3(3)} \dots \xrightarrow{\bar{n}-\Lambda_3(\alpha_3)} \\
& \quad \quad S \cdot \xi(\Lambda_2(1), \dots, \Lambda_2(\alpha_2), \Lambda_3(1), \dots, \Lambda_3(\alpha_3), X - (X_{(2)} \cup X_{(3)})), \quad (13)
\end{aligned}$$

where the  $\Lambda_3(i)$ 's and  $\alpha_3$  depend on  $b_3$  and  $X_{(3)}$ , just in the same way the sets  $\Lambda_2(j)$  and  $\alpha_2$  depend on  $b_2$  and  $X_{(2)}$  and each box has safe-consensus value equal to the value it has in the sequence of (12). We can repeat the very same steps for  $b_4, \dots, b_q$  to obtain the sequence

$$\begin{aligned}
& S \cdot \xi(\Lambda_2(1), \dots, \Lambda_3(\alpha_3), X - (X_{(2)} \cup X_{(3)})) \\
& \quad \xrightarrow{\bar{n}-\Lambda_4(1)} S \cdot \xi(\Lambda_2(1), \dots, \Lambda_3(\alpha_3), \Lambda_4(1), X - (X_{(2)} \cup X_{(3)} \cup \Lambda_4(1))) \\
& \quad \quad \xrightarrow{\bar{n}-\Lambda_4(2)} \dots \xrightarrow{\bar{n}-\Lambda_q(1)} \\
& \quad \quad S \cdot \xi(\Lambda_2(1), \dots, \Lambda_q(1), X - (\Lambda_q(1) \cup \bigcup_{i=2}^{q-1} X_{(i)})) \\
& \quad \quad \quad \xrightarrow{\bar{n}-\Lambda_q(2)} \dots \xrightarrow{\bar{n}-\Lambda_q(\alpha_q)} S \cdot \xi(\Lambda_2(1), \dots, \Lambda_q(\alpha_q), X_{(1)}). \quad (14)
\end{aligned}$$

It is easy to prove that  $L = S \cdot \xi(\Lambda_2(1), \dots, \Lambda_q(\alpha_q), X_{(1)})$  is a ladder state for  $X$  with step  $b_1$  and that each of the sequences of equations (12), (13) and (14) has indistinguishability degree no less than  $n - 2$ . Combining all these sequences, we obtain a new path

$$\mathbf{p}: S_X \sim \dots \sim L$$

with  $\deg \mathbf{p} \geq n - 2$ . □

**Lemma C.3.** *Let  $n \geq 2, X, Y \subseteq \bar{n}$ ,  $\mathcal{A}$  a WOR protocol with safe-consensus objects and  $S$  an state that is reachable in  $\mathcal{A}$  in some round  $r \geq 0$ . Assume also that  $b_j$  is a box representing a safe-consensus object invoked by some processes such that  $b_j \in \mathbf{Inv}(L_1) = \mathbf{Inv}(L_2)$ , where  $L_1 = S \cdot \xi(C_1, \dots, C_u, X_{(j)})$  is a ladder state for  $X$  with step  $b_j$  and  $L_2 = S \cdot \xi(C_1, \dots, C_u, Y_{(j)})$  is a ladder state for  $(X - X_{(j)}) \cup Y_{(j)}$  with step  $b_j$ . Finally, suppose that if  $b_j = \bar{n}$ ,  $\text{scval}(b_j, L_1) = \text{scval}(b_j, L_2)$ . Then  $L_1$  and  $L_2$  are connected in round  $r + 1$  with a  $\mathbf{C}$ -regular path of states  $\mathbf{p}$  that satisfies the following properties:*

- i) *If  $\text{scval}(b_j, L_1) = \text{scval}(b_j, L_2)$  then  $\deg \mathbf{p} \geq n - 2$ .*
- ii) *If  $\text{scval}(b_j, L_1) \neq \text{scval}(b_j, L_2)$  then  $\deg \mathbf{p} \geq n - |b_j|$  and  $\bar{n} - b_j \in \text{iSets}(\mathbf{p})$ .*

*Proof.* Let  $\bar{n}$  be the disjoint union  $\bar{n} = X^- \cup (X \cap Y) \cup Y^- \cup W$ , where

$$X^- = X - Y;$$

$$Y^- = Y - X;$$

$$W = \bar{n} - (X^- \cup (X \cap Y) \cup Y^-).$$

What we need to do to go from  $L_1$  to  $L_2$  is to “interchange”  $X_{(j)}$  with  $Y_{(j)}$ . We first construct a sequence of connected states  $\mathbf{p}_1$  given by

$$\begin{aligned} & S \cdot \xi(C_1, \dots, C_u, X_{(j)}, Y^- \cup W) \\ & \quad \xrightarrow{\bar{n} - \Lambda_j(1)} S \cdot \xi(C_1, \dots, C_u, \Lambda_j(1), X_{(j)} - \Lambda_j(1), Y^- \cup W) \\ & \quad \xrightarrow{\bar{n} - \Lambda_j(2)} S \cdot \xi(C_1, \dots, C_u, \Lambda_j(1), \Lambda_j(2), X_{(j)} - (\Lambda_j(1) \cup \Lambda_j(2)), Y^- \cup W) \\ & \quad \xrightarrow{\bar{n} - \Lambda_j(3)} \dots \xrightarrow{\bar{n} - \Lambda_j(\alpha_j - 1)} S \cdot \xi(C_1, \dots, C_u, \Lambda_j(1), \Lambda_j(2), \dots, \Lambda_j(\alpha_j), Y^- \cup W) \\ & \quad \xrightarrow{\bar{n} - \Lambda_j(\alpha_j)} S \cdot \xi(C_1, \dots, C_u, \Lambda_j(1), \Lambda_j(2), \dots, \Lambda_j(\alpha_j - 1), \Lambda_j(\alpha_j) \cup Y^- \cup W) \\ & \quad \quad \quad \xrightarrow{\bar{n} - \Lambda_j(\alpha_j - 1)} \dots \xrightarrow{\bar{n} - \Lambda_j(1)} S \cdot \xi(C_1, \dots, C_u, X_{(j)} \cup Y^- \cup W), \end{aligned}$$

where the following properties hold:

- $X_{(j)} = \bigcup_{i=1}^{\alpha_j} \Lambda_j(i)$  is a partition of  $X_{(j)}$  such that  $|\Lambda_j(l)| = 1$  for  $l = 2, \dots, \alpha_j$ ;
- $|\Lambda_j(1)| = 2$  if  $|X_{(j)}| > 1$  and  $|\Lambda_j(1)| = |X_{(j)}|$  otherwise;
- $\mathbf{p}_1$  is a  $\mathbf{C}$ -regular sequence (Lemma C.1) with  $\deg \mathbf{p}_1 \geq n - 2$ ;
- the safe-consensus value of every box  $b_i$  is the same in each state of  $\mathbf{p}_1$ . This can be archived by a proper election of elements of the set  $\Lambda_j(1)$  and the Safe-Validity property of the safe-consensus task.



Now, as  $X_{(j)} \cup Y^- = Y_{(j)} \cup X_{(j)}^- \cup (Y^- - Y_{(j)}^-)$ , we can write the state  $S \cdot \xi(C_1, \dots, C_u, X_{(j)} \cup Y^- \cup W)$  as  $S \cdot \xi(C_1, \dots, C_u, Y_{(j)} \cup X_{(j)}^- \cup (Y^- - Y_{(j)}^-) \cup W)$ . We need to build a second path  $\mathbf{p}_2$  as follows:

$$\begin{aligned}
& S \cdot \xi(C_1, \dots, C_u, Y_{(j)} \cup X_{(j)}^- \cup (Y^- - Y_{(j)}^-) \cup W) \\
& \quad \xrightarrow{\bar{n}-Z} S \cdot \xi(C_1, \dots, C_u, \Omega_j(1), (Y_{(j)} - \Omega_j(1)) \cup X_{(j)}^- \cup (Y^- - Y_{(j)}^-) \cup W) \\
& \quad \quad \quad \xrightarrow{\bar{n}-\Omega_j(2)} \dots \xrightarrow{\bar{n}-\Omega_j(\epsilon_j-1)} \\
& \quad \quad \quad S \cdot \xi(C_1, \dots, C_u, \Omega_j(1), \dots, \Omega_j(\epsilon_j) \cup X_{(j)}^- \cup (Y^- - Y_{(j)}^-) \cup W) \\
& \quad \quad \quad \xrightarrow{\bar{n}-\Omega_j(\epsilon_j)} S \cdot \xi(C_1, \dots, C_u, \Omega_j(1), \dots, \Omega_j(\epsilon_j), X_{(j)}^- \cup (Y^- - Y_{(j)}^-) \cup W) \\
& \quad \quad \quad \quad \quad \quad \quad \xrightarrow{\bar{n}-\Omega_j(\epsilon_j-1)} \\
& \quad \quad \quad S \cdot \xi(C_1, \dots, C_u, \Omega_j(1), \dots, \Omega_j(\epsilon_j-1) \cup \Omega_j(\epsilon_j), X_{(j)}^- \cup (Y^- - Y_{(j)}^-) \cup W) \\
& \quad \quad \quad \quad \quad \quad \quad \xrightarrow{\bar{n}-\Omega_j(\epsilon_j-2)} \dots \xrightarrow{\bar{n}-\Omega_j(1)} S \cdot \xi(C_1, \dots, C_u, Y_{(j)}, X_{(j)}^- \cup (Y^- - Y_{(j)}^-) \cup W).
\end{aligned}$$

Let  $L_2$  be the last state of the previous sequence. The next assertions are true for the path  $\mathbf{p}_2$ :

- The sets  $\Omega_j(i)$  and  $\epsilon_j$  are defined for  $Y_{(j)}$  and  $b_j$  in the same way as the  $\Lambda_j(i)$  and  $\alpha_j$  are defined for  $X_{(j)}$  and  $b_j$ ;
- The sequence  $\mathbf{p}_2$  is C-regular (Lemma C.1);
- The safe-consensus value of every box  $c \neq b_j$  is the same in every element of  $\text{States}(\mathbf{p}_2)$ ;
- $\text{scval}(b_j, Q) = \text{scval}(b_j, P)$  for all  $Q, P \in \text{States}(\mathbf{p}_2) - \{R\}$ , where  $R = S \cdot \xi(C_1, \dots, C_u, Y_{(j)} \cup X_{(j)}^- \cup (Y^- - Y_{(j)}^-) \cup W)$ ;
- the set  $Z$  is defined by

$$Z = \begin{cases} b_j & \text{if } \text{scval}(b_j, L_1) \neq \text{scval}(b_j, L_2) \\ \Omega_j(1) & \text{otherwise;} \end{cases}$$

Notice that by the last assertion, we can deduce that  $\deg \mathbf{p}_2 \geq n - |b_j|$  and  $\bar{n} - b_j \in \text{iSets}(\mathbf{p}_2)$  if  $\text{scval}(b_j, L_1) \neq \text{scval}(b_j, L_2)$  and  $\deg \mathbf{p}_2 \geq n - 2$  when  $\text{scval}(b_j, L_1) = \text{scval}(b_j, L_2)$ . Thus we can use  $\mathbf{p}_1$  and  $\mathbf{p}_2$  to obtain a C-regular sequence  $\mathbf{p}$  which fulfills properties i)-ii) of the lemma and that concludes the proof.  $\square$

**Lemma C.4.** *Let  $n \geq 2$ ,  $X, Y \subseteq \bar{n}$ ,  $\mathcal{A}$  a WOR protocol with safe-consensus objects,  $S$  a state that is reachable in  $\mathcal{A}$  in some round  $r \geq 0$  and  $b_j$  a box representing a safe-consensus object invoked by some processes in round  $r + 1$  such that  $b_j \in \text{Inv}(Q_1) = \text{Inv}(Q_2)$ , where  $Q_1 = S \cdot \xi(X)$  and  $Q_2 = S \cdot \xi(Y_{(j)} \cup (X - X_{(j)}))$ . Assume also that if  $b_j = \bar{n}$ ,  $\text{scval}(b_j, Q_1) = \text{scval}(b_j, Q_2)$ . Then the states  $Q_1$  and  $Q_2$  are connected in round  $r + 1$  with a C-regular path of states  $\mathbf{p}$  that satisfies the following properties:*

- If  $\text{scval}(b_j, Q_1) = \text{scval}(b_j, Q_2)$  then  $\deg \mathbf{p} \geq n - 2$ .*
- If  $\text{scval}(b_j, Q_1) \neq \text{scval}(b_j, Q_2)$  then  $\deg \mathbf{p} \geq n - |b_j|$  and  $\bar{n} - b_j \in \text{iSets}(\mathbf{p})$ .*

*Proof.* By Lemma C.2, The state  $Q_1$  can be connected with a state of the form  $Q_{X_{(j)}} = S \cdot \xi(B_1, \dots, B_s, X_{(j)})$  with a C-regular path  $\mathbf{q}_1$  such that  $\deg \mathbf{q}_1 \geq n - 2$ . Using Lemma C.3,  $Q_{X_{(j)}}$  can be connected with  $Q_{Y_{(j)}} = S \cdot \xi(B_1, \dots, B_s, Y_{(j)})$  by means of a C-regular sequence  $\mathbf{q}_2: Q_{X_{(j)}} \sim \dots \sim Q_{Y_{(j)}}$  such that  $\mathbf{q}_2$  satisfies properties i) and ii) of that Lemma. And we can apply Lemma C.2 to connect  $Q_{Y_{(j)}}$  with  $Q_2$  with the C-regular path  $\mathbf{q}_3$  which has indistinguishability degree no less than  $n - 2$ . Therefore the sequence of connected states build by first gluing together the sequences  $\mathbf{q}_1$  and  $\mathbf{q}_2$ , followed by  $\mathbf{q}_3$ , is a C-regular sequence  $\mathbf{q}$  such that the requirements a)-b) are satisfied.  $\square$

To be able to state and prove the last result of this section, we need a definition that will be needed for the rest of the appendix. Let  $Q_1, Q_2$  be two reachable states in round  $r$  of an iterated protocol  $\mathcal{A}$  for  $n$  processes with safe-consensus objects. The set  $\mathfrak{D}_{\mathcal{A}}^r(Q_1, Q_2)$  is defined as

$$\mathfrak{D}_{\mathcal{A}}^r(Q_1, Q_2) = \{b \in \Gamma_{\mathcal{A}}(n) \mid b \in \mathbf{Inv}(Q_1) \cap \mathbf{Inv}(Q_2) \text{ and } \text{scval}(b, Q_1) \neq \text{scval}(b, Q_2)\}.$$

**Theorem C.5.** *Let  $n \geq 2$  and  $X, Y \subseteq \bar{n}$ ,  $\mathcal{A}$  a WOR protocol with safe-consensus objects,  $S$  a reachable state of  $\mathcal{A}$  in some round  $r \geq 0$  and let  $Q_1 = S \cdot \xi(X)$  and  $Q_2 = S \cdot \xi(Y)$  be such that  $\bar{n} \notin \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, Q_2)$ . Then  $Q_1$  and  $Q_2$  are connected in round  $r + 1$  with a C-regular path of states  $\mathbf{p}$  such that*

(A) *If  $\mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, Q_2) = \emptyset$ , then  $\deg \mathbf{p} \geq n - 2$ .*

(B) *If the set  $\mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, Q_2)$  is not empty, then*

1.  $\deg \mathbf{p} \geq \min\{n - |b|\}_{b \in \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, Q_2)}$ ;
2. *for every  $Z \in \mathbf{iSets}(\mathbf{p})$  with  $|Z| < n - 2$ , there exists an unique  $b \in \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, Q_2)$  such that  $Z = \bar{n} - b$ .*

*Proof.* By Lemma C.1,  $\mathfrak{D}_{\mathcal{A}}^r(P_1, P_2) = \{b \in \Gamma_{\mathcal{A}}(n) \mid b \in \mathbf{Inv}_S$  and  $\text{scval}(b, P_1) \neq \text{scval}(b, P_2)\}$ , where  $P_l$  is a one-round successor state of  $S$  and  $\mathbf{Inv}_S = \mathbf{Inv}(P_l)$ ,  $l = 1, 2$ . Let  $\mathbf{Inv}_S = \{b_1, \dots, b_q\}$ . By Lemma C.4, we can connect the state  $Q_1$  with  $R_1 = S \cdot \xi(Y_{(1)} \cup (X - X_{(1)}))$  using a C-regular path  $\mathbf{p}_1: Q_1 \sim \dots \sim R_1$  such that

(A<sub>1</sub>) If  $b_1 \notin \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_1)$ , then  $\deg \mathbf{p}_1 \geq n - 2$ .

(B<sub>1</sub>) If  $b_1 \in \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_1)$ , then  $\deg \mathbf{p}_1 \geq n - |b_1|$  and  $\bar{n} - b_1 \in \mathbf{iSets}(\mathbf{p}_1)$ .

Notice that if there is a set  $Z \in \mathbf{iSets}(\mathbf{p}_1)$  with size strictly less than  $n - 2$ , then it must be true that  $b_1 \in \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_1)$ , because if  $b_1 \notin \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_1)$ , then by (A<sub>1</sub>),  $|Z| \geq n - 2$  and this is impossible. Thus the conclusion of property (B<sub>1</sub>) holds for  $\mathbf{p}_1$ , which means that  $\bar{n} - b_1 \in \mathbf{iSets}(\mathbf{p}_1)$ . Examining the proof of Lemma C.4 we can convince ourselves that every  $W \in \mathbf{iSets}(\mathbf{p}_1)$  such that  $W \neq \bar{n} - b_1$  has cardinality at least  $n - 2$ , gather that,  $Z = \bar{n} - b_1$ .

Applying Lemma C.4 to the states  $R_1$  and  $R_2 = S \cdot \xi(Y_{(1)} \cup Y_{(2)} \cup (X - (X_{(1)} \cup X_{(2)})))$ , we find a C-regular sequence  $\mathbf{p}_2: R_1 \sim \dots \sim R_2$  such that  $\mathbf{p}_2$  and  $b_2$  enjoy the same properties which  $\mathbf{p}_1$  and  $b_1$  have. We can combine the paths  $\mathbf{p}_1$  and  $\mathbf{p}_2$  to obtain a C-regular sequence  $\mathbf{p}_{12}: Q_1 \sim \dots \sim R_2$  from  $Q_1$  to  $R_2$  satisfying the properties

(A<sub>2</sub>) If  $\{b_1, b_2\} \cap \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_2) = \emptyset$ , then  $\deg \mathbf{p}_{12} \geq n - 2$ .

(B<sub>2</sub>) If  $\{b_1, b_2\} \cap \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_2)$  is not empty, then

1.  $\deg \mathbf{p}_{12} \geq \min\{n - |b|\}_{b \in \{b_1, b_2\} \cap \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_2)}$ ;
2. for every  $Z \in \text{iSets}(\mathbf{p}_{12})$  with  $|Z| < n - 2$ , there exists a unique  $b \in \{b_1, b_2\} \cap \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_2)$  such that  $Z = \bar{n} - b$ .

We can repeat this process for all  $s \in \{1, \dots, q\}$ . In general, if  $R_s = S \cdot \xi((\bigcup_i^s Y_{(i)}) \cup (X - (\bigcup_i^s X_{(i)})))$ , we can construct a  $\mathbf{C}$ -regular path

$$\mathbf{p}_{1s}: Q_1 \sim \dots \sim R_s,$$

with the properties

( $A_s$ ) If  $\{b_1, \dots, b_s\} \cap \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_s) = \emptyset$ , then  $\deg \mathbf{p}_{1s} \geq n - 2$ .

( $B_s$ ) If  $\{b_1, \dots, b_s\} \cap \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_s)$  is not empty, then

1.  $\deg \mathbf{p}_{1s} \geq \min\{n - |b|\}_{b \in \{b_1, \dots, b_s\} \cap \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_s)}$ ;
2. for every  $Z \in \text{iSets}(\mathbf{p}_{1s})$  with  $|Z| < n - 2$ , there exists a unique  $b \in \{b_1, \dots, b_s\} \cap \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_s)$  such that  $Z = \bar{n} - b$ .

As  $R_q = S \cdot \xi((\bigcup_i^q Y_{(i)}) \cup (X - (\bigcup_i^q X_{(i)}))) = S \cdot \xi(Y) = Q_2$ , the sequence  $\mathbf{p}_{1q}$  is the desired  $\mathbf{C}$ -regular path from  $Q_1$  to  $Q_2$ , fulfilling conditions ( $A$ ) and ( $B$ ) (because  $\{b_1, \dots, b_q\} \cap \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, R_q) = \text{Inv}_S \cap \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, Q_2) = \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_1, Q_2)$ ). The result follows.  $\square$

## C.2 The main structural results

In this section, we prove Lemma C.8. For a given path  $\mathfrak{s}$  of connected states of an iterated protocol  $\mathcal{A}$ , consider the set

$$\beta_{\mathcal{A}}(\mathfrak{s}) = \{b \in \Gamma_{\mathcal{A}}(n) \mid (\exists X, Y \in \text{iSets}(\mathfrak{s}))(|X \cap b| = 1 \text{ and } |Y \cap b| = 1)\},$$

and if  $m \in \bar{n}$ , let  $\beta_{\mathcal{A}}(\mathfrak{s}; m) = \beta_{\mathcal{A}}(\mathfrak{s}) \cap \Gamma_{\mathcal{A}}(n, m)$ . Notice that  $\beta_{\mathcal{A}}(\mathfrak{s}; m) \subseteq V_{n, m}$  (see Section B.1).

**Lemma C.6.** *Let  $n \geq 3$  and  $1 \leq v \leq s \leq n - 2$  be fixed. Suppose that  $\mathcal{A}$  is a WOR protocol with safe-consensus objects and there exists a sequence*

$$\mathfrak{s}: S_0 \stackrel{X_1}{\sim} \dots \stackrel{X_q}{\sim} S_q \quad (q \geq 1),$$

*of connected states of round  $r \geq 0$  such that*

$$\Lambda_1 \quad \deg \mathfrak{s} \geq v.$$

$$\Lambda_2 \quad \text{For every } X_i \text{ with } v \leq |X_i| < s, \text{ there exists } b_i \in \Gamma_{\mathcal{A}}(n, n - |X_i|) \text{ such that } X_i = \bar{n} - b_i$$

$$\Lambda_3 \quad \bar{n} \notin \beta(\mathfrak{s}).$$

*Then in round  $r + 1$  there exists a sequence*

$$\mathfrak{q}: Q_0 \sim \dots \sim Q_u,$$

*of connected successor states of all the  $S_j$ , such that the following statements hold:*

$$\Psi_1 \quad \text{If } \beta_{\mathcal{A}}(\mathfrak{s}; n - v + 1) = \emptyset, \text{ then } \deg \mathfrak{q} \geq v.$$

$\Psi_2$  If  $\beta_{\mathcal{A}}(\mathfrak{s}; n - v + 1)$  is not empty, then  $\deg \mathfrak{q} \geq v - 1$ .

$\Psi_3$  For every  $Z \in \text{iSets}(\mathfrak{q})$  with  $|Z| = v - 1$ , there exist  $X, X' \in \text{iSets}(\mathfrak{s})$  and a unique  $b \in \beta_{\mathcal{A}}(\mathfrak{s}; n - v + 1)$  such that  $Z = \bar{n} - b = X \cap X'$  and  $b = c_1 \cup c_2$ ,  $c_k \in \Gamma_{\mathcal{A}}(n, n - v)$ .

$\Psi_4$  For every  $Z \in \text{iSets}(\mathfrak{q})$  with  $v \leq |Z| < s$ , there exists a unique  $b \in \Gamma_{\mathcal{A}}(n, n - |Z|)$  such that  $Z = \bar{n} - b$ .

$\Psi_5$   $\beta_{\mathcal{A}}(\mathfrak{q}; n - l + 1) \subseteq \zeta(\beta_{\mathcal{A}}(\mathfrak{s}; n - l))$  for  $v \leq l < s$ .

*Proof.* In order to find the path  $\mathfrak{q}$  that has the properties  $\Psi_1$ - $\Psi_5$ , first we need to define a set of states of round  $r + 1$  of  $\mathcal{A}$ , called  $\Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$ , which we will use to construct the path  $\mathfrak{q}$ . The key ingredient of  $\Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  is the safe-consensus values of the boxes used in each member of  $\Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$ . Define the set of states  $\Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  as

$$\Phi_{\mathcal{A}}^{r+1}(\mathfrak{s}) = \{R \mid R = S \cdot \xi(X) \text{ and } (S, X) \in \text{States}(\mathfrak{s}) \times \text{iSets}(\mathfrak{s})\}.$$

Each element of  $\Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  is a state in round  $r + 1$  of  $\mathcal{A}$  which is obtained when all the processes with ids in some set  $X \in \text{iSets}(\mathfrak{s})$  execute concurrently the operations of  $\mathcal{A}$ , followed by all processes with ids in  $\bar{n} - X$ . The safe-consensus values of each box for the states of  $\Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  in round  $r + 1$  are defined by using the following rules: Let  $b$  be any box such that  $b \notin \beta(\mathfrak{s})$ .

- If  $|X \cap b| \neq 1$  for every  $X \in \text{iSets}(\mathfrak{s})$ , then we claim that we can choose any element  $j$  such that  $j$  is the safe-consensus value of  $b$  in every state  $R \in \Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  such that  $b \in \mathbf{Inv}(R)$ .
- If there exists exactly one set  $X \in \text{iSets}(\mathfrak{s})$  such that  $X \cap b = \{x\}$ , then  $b$  has  $x$  as its safe-consensus value in every state  $Q \in \Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  with  $b \in \mathbf{Inv}(Q)$ .

We establish rules to define the safe-consensus values of every element of the set  $\beta_{\mathcal{A}}(\mathfrak{s})$ , using the order on the set  $\text{iSets}(\mathfrak{s})$  induced by the path  $\mathfrak{s}$ , when we traverse  $\mathfrak{s}$  from  $S_0$  to  $S_q$ . So that  $\text{iSets}(\mathfrak{s})$  is ordered as

$$X_1, \dots, X_q. \tag{15}$$

For each  $b \in \beta_{\mathcal{A}}(\mathfrak{s})$ , suppose that  $X_i, X_{i+z_1}, \dots, X_{i+z_k}$ , ( $1 \leq i < i + z_1 < i + z_2 < \dots < i + z_k \leq q$ ;  $z_j > 0$  and  $k \geq 1$ ) is the (ordered) subset of  $\text{iSets}(\mathfrak{s})$  of all  $X \in \text{iSets}(\mathfrak{s})$  with the property  $|X \cap b| = 1$ . Take the set  $X_i$ . If  $x_i \in X_i \cap b$ , then  $x_i$  is the safe-consensus value of  $b$  for all the elements of  $\Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  of the form  $P_j = S \cdot \xi(X_j)$  where  $b \in \mathbf{Inv}(P_j)$  and  $1 \leq j \leq i + z_1 - 1$ . Next, in all states  $R_u \in \Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  such that  $R_u = S \cdot \xi(X_u)$  with  $b \in \mathbf{Inv}(R_u)$  and  $i + z_1 \leq u \leq i + z_2 - 1$ ,  $b$  has safe-consensus value equal to  $x_{i+z_1} \in X_{i+z_1} \cap b$ . In general, in all states  $T_v \in \Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  of the form  $T_v = S \cdot \xi(X_v)$  where  $b \in \mathbf{Inv}(T_v)$  and  $i + z_l \leq v \leq i + z_{l+1} - 1$  and  $1 \leq l \leq k - 1$ ,  $b$  has safe-consensus value equal to  $x_{i+z_l} \in X_{i+z_l} \cap b$ . Finally, in each state  $L_w = S \cdot \xi(X_w) \in \Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  with  $b \in \mathbf{Inv}(L_w)$  and  $i + z_k \leq w \leq q$ ,  $b$  has safe-consensus value equal to  $x_{i+z_k} \in X_{i+z_k} \cap b$ .

Using the Safe-Validity property of the safe-consensus task, it is an easy (but long) routine task to show that we can find states of  $\mathcal{A}$  in round  $r + 1$  which have the form of the states given in  $\Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  and with the desired safe-consensus values for every box.

We are ready to build the sequence  $\mathfrak{q}$  of the Lemma<sup>2</sup>. We use induction on  $q \geq 1$ . For the base case when  $q = 1$ , we have that  $\mathfrak{s}: S_0 \stackrel{X_1}{\sim} S_1$ , with  $|X_1| \geq v$ . Here we easily build the sequence

---

<sup>2</sup>Notice that the order given to  $\Phi_{\mathcal{A}}^{r+1}(\mathfrak{s})$  in equation 15 is the precise order in which the states of this set will appear in the path  $\mathfrak{q}$ .

$S_0 \cdot \xi(X_1) \stackrel{X_1}{\sim} S_1 \cdot \xi(X_1)$  which clearly satisfies properties  $\Psi_1$ - $\Psi_5$ . Assume that for  $1 \leq w < q$  we have a sequence

$$\mathbf{q}': Q_0 \sim \dots \sim Q_l \quad (l \geq 1),$$

Satisfying conditions  $\Psi_1$ - $\Psi_5$  and such that  $Q_0 = S_0 \cdot \xi(X_1)$ ,  $Q_l = S_w \cdot \xi(X_w)$ . We now connect the state  $Q_l$  with  $Q = S_{w+1} \cdot \xi(X_{w+1})$ . By Theorem C.5, there is a  $\mathbf{C}$ -regular sequence

$$\mathbf{v}: Q_l \sim \dots \sim Q'$$

such that  $Q' = S_w \cdot \xi(X_{w+1})$  and  $\mathbf{v}, Q_l$  and  $Q'$  satisfy conditions (A)-(B) of that lemma. Clearly  $\mathfrak{D}_{\mathcal{A}}^{r+1}(Q_l, Q') \subseteq \beta_{\mathcal{A}}(\mathfrak{s})$ , because the only boxes used by  $\mathcal{A}$  in the states  $Q_l$  and  $Q'$  with different safe-consensus value, are the boxes which intersect two different sets  $X, X' \in \text{iSets}(\mathfrak{s})$  in one element.

Joining the sequence  $\mathbf{q}'$  with  $\mathbf{v}$ , followed by the small path  $\mathbf{t}: Q' \stackrel{X_{w+1}}{\sim} Q$  (of degree at least  $v$ ), we obtain a new sequence  $\mathbf{q}: Q_0 \sim \dots \sim Q$ . We now need to show that  $\mathbf{q}$  satisfies properties  $\Psi_1$ - $\Psi_5$ .

$\Psi_1$  Notice that every box in  $\beta(\mathfrak{s})$  has size no bigger than  $n - v + 1$ . Suppose that  $\beta_{\mathcal{A}}(\mathfrak{s}; n - v + 1) = \emptyset$ . In particular, this implies that  $\mathfrak{D}_{\mathcal{A}}^{r+1}(Q_l, Q') \cap \Gamma_{\mathcal{A}}(n, n - v + 1) = \emptyset$ . If it happens that  $\mathfrak{D}_{\mathcal{A}}^{r+1}(Q_l, Q')$  is void, then as condition (A) of Theorem C.5 holds for  $\mathbf{v}$ ,  $\deg \mathbf{v} \geq v$ . But if  $\mathfrak{D}_{\mathcal{A}}^{r+1}(Q_l, Q') \neq \emptyset$  we have that  $|b| \leq n - v$  for all  $b \in \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_l, Q')$ , thus by part 1. of property (B) of Theorem C.5,

$$\deg \mathbf{v} \geq \min\{n - |b|\}_{b \in \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_l, Q')} \geq v.$$

By the induction, hypothesis  $\deg \mathbf{q}' \geq v$  and also  $\deg \mathbf{t} \geq v$ , so that  $\deg \mathbf{q} \geq v$ .

$\Psi_2$  If  $\beta_{\mathcal{A}}(\mathfrak{s}; n - v + 1) \neq \emptyset$ , then either  $\deg \mathbf{q}' \geq v - 1$  (by the induction hypothesis) or  $\deg \mathbf{v} \geq v - 1$ . This last assertion is true, because by (B) of Theorem C.5, we have that  $\deg \mathbf{v} \geq \min\{n - |b|\}_{b \in \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_l, Q')} \geq v - 1$ . Gather that,  $\deg \mathbf{q} \geq v - 1$ .

$\Psi_3$  We remark first that  $\text{iSets}(\mathbf{q}) = \text{iSets}(\mathbf{q}') \cup \text{iSets}(\mathbf{v}) \cup \text{iSets}(\mathbf{t})$ . If we are given  $Z \in \text{iSets}(\mathbf{q})$  such that  $|Z| = v - 1$ , then  $Z \in \text{iSets}(\mathbf{q}')$  or  $Z \in \text{iSets}(\mathbf{v})$ . When  $Z \in \text{iSets}(\mathbf{q}')$ , we use our induction hypothesis to show that  $Z = Y \cap Y' = \bar{n} - d$  for some  $Y, Y' \in \text{iSets}(\mathfrak{s})$  and unique  $d \in \beta_{\mathcal{A}}(\mathfrak{s}; n - v + 1)$  such that  $d = d' \cup d''$ ,  $d', d'' \in \Gamma_{\mathcal{A}}(n, n - v)$ . On the other hand, if  $Z \in \text{iSets}(\mathbf{v})$ , it must be true that  $\mathfrak{D}_{\mathcal{A}}^{r+1}(Q_l, Q') \neq \emptyset$  (if not, then  $\deg \mathbf{v} \geq n - 2$  by property (A) of Theorem C.5 for  $\mathbf{v}$  and this contradicts the size of  $Z$ ). As  $v - 1 < n - 2$ , we can apply part 2 of condition (B) of Theorem C.5 to deduce that  $Z = \bar{n} - b$  for a unique  $b \in \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_l, Q')$ . Because  $|Z| = v - 1$ ,  $|b| = n - v + 1$ . Now,  $b \in \mathfrak{D}_{\mathcal{A}}^{r+1}(Q_l, Q') \subseteq \beta_{\mathcal{A}}(\mathfrak{s})$ , gather that, there exists  $X, X' \in \text{iSets}(\mathfrak{s})$  such that  $|X \cap b| = 1$  and  $|X' \cap b| = 1$  and this clearly implies that  $|X| = |X'| = v$  and  $\bar{n} - b = X \cap X'$ , that is,

$$Z = \bar{n} - b = X \cap X',$$

and finally, we apply  $\Lambda_2$  of the sequence  $\mathfrak{s}$  to  $X, X'$  to find two unique boxes  $c, c' \in \Gamma_{\mathcal{A}}(n, n - v)$  with  $X = \bar{n} - c$  and  $X' = \bar{n} - c'$ , so that

$$\bar{n} - b = X \cap X' = (\bar{n} - c) \cap (\bar{n} - c') = \bar{n} - (c \cup c'),$$

then  $b = c \cup c'$  and condition  $\Psi_3$  is satisfied by  $\mathbf{q}$ .

$\Psi_4$  The sequence  $\mathbf{q}$  fulfills this property because of the induction hypothesis on  $\mathbf{q}'$ , condition  $\Lambda_2$  for  $\mathfrak{s}$  and (B) of Theorem C.5 for  $\mathbf{v}$ .

$\Psi_5$  Let  $l \in \{v, \dots, s\}$ . This property is satisfied by  $\mathbf{q}$  if  $\beta_{\mathcal{A}}(\mathbf{q}; n - l + 1) = \emptyset$ . Suppose that  $c \in \beta_{\mathcal{A}}(\mathbf{q}; n - l + 1)$ , there exist  $X, Y \in \text{iSets}(\mathbf{q})$  such that  $|X \cap c| = |Y \cap c| = 1$ . Then  $|X| = |Y| = l$ , so that we use property  $\Psi_3$  (or  $\Psi_4$ ) on  $X, Y$  to find two boxes  $b_X, b_Y \in \beta_{\mathcal{A}}(\mathbf{s}; n - l)$  such that  $X = \bar{n} - b_X$  and  $Y = \bar{n} - b_Y$ . Also,  $X \cap Y = \bar{n} - c$ , thus  $\bar{n} - c = X \cap Y = (\bar{n} - b_X) \cap (\bar{n} - b_Y) = \bar{n} - (b_X \cup b_Y)$ . Therefore  $c = b_X \cup b_Y$  and this says that  $c \in \zeta(\beta_{\mathcal{A}}(\mathbf{s}; n - l))$ . Condition  $\Psi_5$  is fulfilled by  $\mathbf{q}$ .

We have shown with induction that we can build the path  $\mathbf{q}$  from the sequence  $\mathbf{s}$ , no matter how many states  $\mathbf{s}$  has. This proves the Lemma.  $\square$

The following corollary is a weaker version of Lemma C.6, and it can be proven mostly as a consequence of that result.

**Corollary C.7.** *Let  $n \geq 3$  and  $1 \leq s \leq n - 2$  be fixed. Suppose that  $\mathcal{A}$  is a WOR protocol with safe-consensus objects and there exists a sequence*

$$\mathbf{s}: S_0 \stackrel{X_1}{\sim} \dots \stackrel{X_r}{\sim} S_q \quad (q \geq 1),$$

*of connected states of round  $r \geq 0$  such that  $\deg \mathbf{s} \geq s$  and  $\bar{n} \notin \beta_{\mathcal{A}}(\mathbf{s})$ . Then in round  $r + 1$  there exists a sequence*

$$\mathbf{q}: Q_0 \stackrel{Z_1}{\sim} \dots \stackrel{Z_u}{\sim} Q_u,$$

*such that the following statements hold:*

$\Psi_1$  *If  $\beta_{\mathcal{A}}(\mathbf{s}; n - s + 1) = \emptyset$ , then  $\deg \mathbf{q} \geq s$ .*

$\Psi_2$  *If  $\beta_{\mathcal{A}}(\mathbf{s}; n - s + 1)$  is not empty, then  $\deg \mathbf{q} \geq s - 1$ .*

$\Psi_3$  *For every  $Z \in \text{iSets}(\mathbf{q})$  with  $|Z| = s - 1$ , there exist  $X, X' \in \text{iSets}(\mathbf{s})$  and a unique  $b \in \beta_{\mathcal{A}}(\mathbf{s}; n - s + 1)$  such that  $Z = \bar{n} - b = X \cap X'$ .*

$\Psi_5$   $\beta_{\mathcal{A}}(\mathbf{q}; n - s + 2) \subseteq \zeta(\beta_{\mathcal{A}}(\mathbf{s}; n - s + 1))$ .

We have gathered all the required elements to prove one of the key ingredient of the full proof of Theorem 4.2, it is the following

**Lemma C.8.** *Let  $n \geq 3$  and  $3 \leq m \leq n$  be fixed. Suppose that  $\mathcal{A}$  is a WOR protocol with safe-consensus objects such that  $\nu_{\mathcal{A}}(n, m) \leq n - m$ . If  $S^r, Q^r$  are two reachable states in  $\mathcal{A}$  for some round  $r \geq 0$ , connected with a sequence  $\mathbf{q}: S^r \sim \dots \sim Q^r$  such that  $\deg \mathbf{q} \geq n - m + 1$ , then for all  $u \geq 0$ , there exist successor states  $S^{r+u}, Q^{r+u}$  of  $S^r$  and  $Q^r$  respectively, in round  $r + u$  of  $\mathcal{A}$ , such that  $S^{r+u}$  and  $Q^{r+u}$  are connected.*

*Proof.* Let  $\mathcal{A}$  be a protocol with the given hypothesis, set  $\mathbf{q}_0 = \mathbf{q}$  and  $m = n - s + 1$ . Because  $3 \leq m \leq n$ ,  $s \in \{1, \dots, n - 2\}$ . Using the sequence  $\mathbf{q}_0$  and applying Corollary C.7, we build a path  $\mathbf{q}_1: S^{r+1} \sim \dots \sim Q^{r+1}$ , connecting the successor states  $S^{r+1}, Q^{r+1}$  of  $S^r$  and  $Q^r$  respectively, such that

$\Psi_{1,1}$  *If  $\beta_{\mathcal{A}}(\mathbf{q}_0; n - s + 1) = \emptyset$ , then  $\deg \mathbf{q}_1 > s - 1$ .*

$\Psi_{1,2}$  *If  $\beta_{\mathcal{A}}(\mathbf{q}_0; n - s + 1)$  is not empty, then  $\deg \mathbf{q}_1 \geq s - 1$ .*

$\Psi_{1,3}$  For every  $Z \in \text{iSets}(\mathbf{q}_1)$  with  $|Z| = s - 1$ , there exist  $X, X' \in \text{iSets}(\mathbf{q}_0)$  and a unique  $b \in \beta_{\mathcal{A}}(\mathbf{q}_0; n - s + 1)$  such that  $Z = \bar{n} - b = X \cap X'$ .

$\Psi_{1,5}$   $\beta_{\mathcal{A}}(\mathbf{q}_1; n - s + 2) \subseteq \zeta(\beta_{\mathcal{A}}(\mathbf{q}_0; n - s + 1))$ .

Starting from  $\mathbf{q}_1$  and using induction together with Lemma C.6, we can prove that for each  $u \in \{1, \dots, s - 1\}$ , there exist successor states  $S^{r+u}, Q^{r+u}$  of the states  $S^r$  and  $Q^r$  respectively, and a sequence

$$\mathbf{q}_u: S^{r+u} \sim \dots \sim Q^{r+u},$$

satisfying the properties:

$\Psi_{u,1}$  If  $\beta_{\mathcal{A}}(\mathbf{q}_{u-1}; n - s + u) = \emptyset$ , then  $\deg \mathbf{q}_u > s - u$ .

$\Psi_{u,2}$  If  $\beta_{\mathcal{A}}(\mathbf{q}_{u-1}; n - s + u)$  is not empty, then  $\deg \mathbf{q}_u \geq s - u$ .

$\Psi_{u,3}$  For every  $Z \in \text{iSets}(\mathbf{q}_u)$  with  $|Z| = s - 1$ , there exist  $X, X' \in \text{iSets}(\mathbf{q}_{u-1})$  and a unique  $b \in \beta_{\mathcal{A}}(\mathbf{q}_{u-1}; n - s + 1)$  such that  $Z = \bar{n} - b = X \cap X'$ .

$\Psi_{u,4}$  For every  $Z \in \text{iSets}(\mathbf{q}_u)$  with  $s - u \leq |Z| < s - 1$ , there exist  $X, X' \in \text{iSets}(\mathbf{q}_{u-1})$  and a unique  $b \in \beta_{\mathcal{A}}(\mathbf{q}_{u-1}; n - |Z|)$  such that  $Z = \bar{n} - b = X \cap X'$  and  $b = c_1 \cup c_2$ ,  $c_k \in \beta_{\mathcal{A}}(\mathbf{q}_{u-1}; n - |Z| - 1)$ .

$\Psi_{u,5}$   $\beta_{\mathcal{A}}(\mathbf{q}_u; n - l + 1) \subseteq \zeta(\beta_{\mathcal{A}}(\mathbf{q}_{u-1}; n - l))$  for  $s - u \leq l < s$ .

When  $u = s - 1$ , we obtain a sequence  $\mathbf{q}_{s-1}: S^{r+s-1} \sim \dots \sim Q^{r+s-1}$  that connects the states  $S^{r+s-1}$  and  $Q^{r+s-1}$ , such that

$\Psi_{s-1,1}$  If  $\beta_{\mathcal{A}}(\mathbf{q}_{s-2}; n - 1) = \emptyset$ , then  $\deg \mathbf{q}_{s-1} > 1$ .

$\Psi_{s-1,2}$  If  $\beta_{\mathcal{A}}(\mathbf{q}_{s-2}; n - 1)$  is not empty, then  $\deg \mathbf{q}_{s-1} \geq 1$ .

$\Psi_{s-1,3}$  For every  $Z \in \text{iSets}(\mathbf{q}_{s-1})$  with  $|Z| = s - 1$ , there exist  $X, X' \in \text{iSets}(\mathbf{q}_{s-2})$  and a unique  $b \in \beta_{\mathcal{A}}(\mathbf{q}_{s-2}; n - s + 1)$  such that  $Z = \bar{n} - b = X \cap X'$ .

$\Psi_{s-1,4}$  For every  $Z \in \text{iSets}(\mathbf{q}_{s-1})$  with  $1 \leq |Z| < s - 1$ , there exist  $X, X' \in \text{iSets}(\mathbf{q}_{s-2})$  and a unique  $b \in \beta_{\mathcal{A}}(\mathbf{q}_{s-2}; n - |Z|)$  such that  $Z = \bar{n} - b = X \cap X'$  and  $b = c_1 \cup c_2$ ,  $c_k \in \beta_{\mathcal{A}}(\mathbf{q}_{s-2}; n - |Z| - 1)$ .

$\Psi_{s-1,5}$   $\beta_{\mathcal{A}}(\mathbf{q}_{s-1}; n - l + 1) \subseteq \zeta(\beta_{\mathcal{A}}(\mathbf{q}_{s-2}; n - l))$  for  $1 \leq l < s$ .

Our final goal is to show that for all  $v \geq s - 1$ , we can connect in each round  $r + v$  of  $\mathcal{A}$ , successor states of  $S^r$  and  $Q^r$ . We first claim that for any  $w = 0, \dots, s - 1$  and  $z \geq 0$ ,

$$\zeta^z(\beta_{\mathcal{A}}(\mathbf{q}_w; n - s + 1)) \subseteq \zeta^z(\Gamma_{\mathcal{A}}(n, n - s + 1)),$$

(this is true because  $\zeta$  preserves  $\subseteq$ ) and combining this fact with the properties  $\Psi_{u,5}$  and induction, we can show that for  $1 \leq l < s$

$$\beta_{\mathcal{A}}(\mathbf{q}_{s-1}; n - l + 1) \subseteq \zeta^{s-l}(\beta_{\mathcal{A}}(\mathbf{q}_{l-1}; n - s + 1)) \subseteq \zeta^{s-l}(\Gamma_{\mathcal{A}}(n, n - s + 1)). \quad (16)$$

And because  $|\Gamma_{\mathcal{A}}(n, n - s + 1)| = \nu_{\mathcal{A}}(n, n - s + 1) \leq s - 1$  and  $m = n - s + 1$ , we use Theorem B.7 to check that  $\zeta^{s-1}(\Gamma_{\mathcal{A}}(n, n - s + 1)) = \emptyset$ , implying that

$$\beta_{\mathcal{A}}(\mathbf{q}_{s-1}; n) = \emptyset.$$

With all this data, Lemma C.6 and an easy inductive argument (starting at the base case  $v = s - 1$ ), we can find for all  $v \geq s - 1$ , states  $S^{r+v}, Q^{r+v}$  of round  $r + v$  of  $\mathcal{A}$ , which are successor states of  $S^r$  and  $Q^r$  respectively and connected with a sequence  $\mathbf{q}_v$  such that

$$\Psi'_{v,1} \text{ deg } \mathbf{q}_v \geq 1.$$

$$\Psi'_{v,2} \text{ For every } Z \in \text{iSets}(\mathbf{q}_v) \text{ with } |Z| = s - 1, \text{ there exist } X, X' \in \text{iSets}(\mathbf{q}_{v-1}) \text{ and a unique } b \in \beta_{\mathcal{A}}(\mathbf{q}_{v-1}; n - s + 1) \text{ such that } Z = \bar{n} - b = X \cap X'.$$

$$\Psi'_{v,3} \text{ For every } Z \in \text{iSets}(\mathbf{q}_v) \text{ with } 1 \leq |Z| < s - 1, \text{ there exist } X, X' \in \text{iSets}(\mathbf{q}_{v-1}) \text{ and a unique } b \in \beta_{\mathcal{A}}(\mathbf{q}_{v-1}; n - |Z|) \text{ such that } Z = \bar{n} - b = X \cap X' \text{ and } b = c_1 \cup c_2, c_k \in \beta_{\mathcal{A}}(\mathbf{q}_{v-1}; n - |Z| - 1).$$

$$\Psi'_{v,4} \beta_{\mathcal{A}}(\mathbf{q}_v; n - l + 1) \subseteq \zeta^{s-l}(\Gamma_{\mathcal{A}}(n, n - s + 1)) \text{ for } 1 \leq l < s.$$

It is precisely these four properties of the path  $\mathbf{q}_v$  which allow us to find the path  $\mathbf{q}_{v+1}$ , applying Lemma C.6, such that  $\mathbf{q}_{v+1}$  enjoys the same properties. Therefore, starting from round  $r$ , we can connect in all rounds of  $\mathcal{A}$ , successor states of  $S^r$  and  $Q^r$ . The Lemma is proven.  $\square$

### C.3 The case $\nu_{\mathcal{A}}(n, 2) \leq n - 2$

The last ingredient that we need to prove Theorem 4.2, is Lemma C.10. This can be done easily using the following result.

**Lemma C.9.** *Let  $\mathcal{A}$  be a WOR protocol with safe-consensus objects. Suppose that there exists a partition  $\bar{n} = A \cup B$  and a sequence*

$$\mathbf{p}: S_0 \sim \dots \sim S_l \quad (l \geq 0)$$

*of connected states in round  $r \geq 0$  of  $\mathcal{A}$ , with the following properties*

$$I) \text{ iSets}(\mathbf{p}) = \{A, B\};$$

$$II) (\forall b \in \Gamma_{\mathcal{A}}(n, 2))(b \subseteq A \text{ or } b \subseteq B).$$

*Then in round  $r + 1$  of  $\mathcal{A}$  there exists a path*

$$\mathbf{q}: Q_0 \sim \dots \sim Q_s \quad (s \geq 1)$$

*of connected states and the following properties hold:*

$$a) \text{ Each state } Q_k \text{ is of the form } Q_k = S_j \cdot \xi(X), \text{ where } X = A \text{ or } X = B;$$

$$b) \text{ iSets}(\mathbf{q}) = \{A, B\}.$$

*Proof.* This proof is very similar in spirit to the proof of Lemma C.6, so we omit it.  $\square$

**Lemma C.10.** *Let  $n \geq 2$ . If  $\mathcal{A}$  is a WOR protocol for  $n$  processes using safe-consensus objects with  $\nu_{\mathcal{A}}(n, 2) \leq n - 2$  and  $S$  is a reachable state in  $\mathcal{A}$  for some round  $r \geq 0$ , then there exists a partition of the set  $\bar{n} = A \cup B$  such that for all  $u \geq 0$ , the states  $S \cdot \xi^u(A)$  and  $S \cdot \xi^u(B)$  are connected.*

*Proof.* This proof is analogous to the proof of Lemma C.8. We use Lemma B.3 to find the partition of  $\bar{n}$  and then we apply induction combined with Lemma C.9. We omit the details.  $\square$



## C.4 The proof of Theorem 4.2

Here we give the proof of Theorem 4.2 for any  $n \geq 2$ , thus completing all the necessary proofs of the paper.

**Proof of Theorem 4.2** Assume that there exists a protocol  $\mathcal{A}$  for consensus such that there is some  $m$  with  $2 \leq m \leq n$  with  $\nu_{\mathcal{A}}(n, m) \leq n - m$ . Let  $O, U$  be the initial states in which all processes have as input values 0s and 1s respectively. We now find successor states of  $O$  and  $U$  in each round  $r \geq 0$ , which are connected. We have cases:

*Case  $m = 2$ .* By Lemma C.10, there exists a partition of  $\bar{n} = A \cup B$  such that for any state  $S$  and any  $r \geq 0$ ,  $S \cdot \xi^r(A)$  and  $S \cdot \xi^r(B)$  are connected. Let  $OU$  be the initial state in which all processes with ids in  $A$  have as input value 0s and all processes with ids in  $B$  have as input values 1s. Then for all  $r \geq 0$  we have that

$$O \cdot \xi^r(A) \stackrel{A}{\sim} OU \cdot \xi^r(A) \quad \text{and} \quad OU \cdot \xi^r(B) \stackrel{B}{\sim} U \cdot \xi^r(B)$$

and by Lemma C.10, the states  $OU \cdot \xi^r(A)$  and  $OU \cdot \xi^r(B)$  are connected. Thus, for any  $r$ -round partial execution of  $\mathcal{A}$ , we can connect the states  $O^r = O \cdot \xi^r(A)$  and  $U^r = U \cdot \xi^r(B)$ .

*Case  $3 \leq m \leq n$ .* By Lemma 3.2, we know that any two initial states for consensus are connected, so that we can connect  $O$  and  $U$  with a sequence  $\mathbf{q}$  of initial states of  $\mathcal{A}$  and it is not hard to check that  $\deg \mathbf{q} \geq n - 1 \geq n - m + 1$ . By Lemma C.8, for each round  $r \geq 0$  of  $\mathcal{A}$ , there exist successor states  $O^r, U^r$  of  $O$  and  $U$  respectively, such that  $O^r$  and  $U^r$  are connected.

In this way, we have connected successor states of  $O$  and  $U$  in each round of the protocol  $\mathcal{A}$ . Now,  $O$  is a 0-valent, initial state, which is connected to the initial state  $U$ , so that we can apply Lemma 3.3 to conclude that  $U$  is 0-valent. But this contradicts the fact that  $U$  is a 1-valent state, so we have reached a contradiction. Therefore  $\nu_{\mathcal{A}}(n, m) > n - m$ .  $\square$